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 feature that simplifies interacting with databases in Laravel applications. It provides a fluent, chainable interface to build and execute database queries. The query builder abstracts the underlying database, supports multiple database systems, and allows you to write database-agnostic code. It offers a simple and elegant syntax, similar to SQL, making it easier to construct queries. The query builder provides additional features such as parameter binding, query logging, and transaction support, and integrates seamlessly with Laravel's ORM. Overall, it offers a convenient and expressive way to work with databases.

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:

```
Function questiontwo(){

$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 fetch only unique values from a column or set of columns in a database table. It ensures that duplicate values are excluded, and only distinct values are returned. When combined with the select() method, it filters the query results to include only unique values from the specified columns. It is helpful when you need to eliminate duplicates or count distinct values in your query results. However, it should be used thoughtfully, as it alters the result set to contain unique values only.

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:

```
Function questionfour(){

$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:

```
Function questionfive(){

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

foreach ($posts as $post) {

   echo $post;

}

}
```

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() method is used when you want to retrieve the first record that matches certain conditions. You can use the where() method to specify the conditions for the query. It returns the first matching record or null if no record is found.

```
$test1 = DB::table('table_name') ->where('column', 'value')->first();
```

The find() method is used when you know the primary key value of the record you want to retrieve. You pass the primary key value as an argument to the find() method, and it fetches the record with the corresponding primary key value. If no record is found, it returns null.

```
$test2 = DB::table('table_name') ->find($id);
```

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:

```
Function questionseven(){

$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:

```
$Eight = DB::table('posts')->insert([

    'title' => 'X',

    'slug' => 'X',

    'excerpt' => 'excerpt',

    'description' => 'description',
    'is_published' => true,
    'min_to_read' => 2,
]);

echo $Eight;
```

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:

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

echo $Nine;
```

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 row.

Answer:

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

```
 ->delete();

echo $Ten;
```

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(): Returns the number of records that match the query.

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

echo $count;
```

sum(): Calculates the sum of a column's values.

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

echo $totalAmount;
```

avg(): Calculates the average value of a column.

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

echo $averagePrice;
```

max(): Retrieves the maximum value from a column.

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

```
echo $highestScore;
```

min(): Retrieves the minimum value from a column.

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

echo $lowestTemperature;
```

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 a query. It allows you to retrieve records where a specific column's value is not equal to the provided value.

```
$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?

Answer:

exists(): The exists() method is used to check if any records match the query. It returns true if at least one record exists, and false otherwise.

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

doesntExist(): The doesntExist() method is used to check if no records match the query. It returns true if no records exist, and false if there is at least one matching record.

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

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:

```
$fifteen = DB::table('posts') ->where('id', 3)
            ->increment('min_to_read', 1);


echo $fifteen;
```