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 simple and elegant way to interact with databases. It allows developers to build database queries using a fluent, chainable interface, instead of writing raw SQL queries. The query builder abstracts the underlying database system, making it easier to write database queries that are independent of the specific database engine being used.

The query builder provides methods for constructing various types of queries, including selecting, inserting, updating, and deleting records. It also supports advanced features like joins, conditional clauses, ordering, grouping, and aggregations. The query builder takes care of generating the appropriate SQL syntax based on the method calls, making it easier to work with databases in Laravel.

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: To retrieve the "excerpt" and "description" columns from the "posts" table using Laravel's query builder and store the result in the $posts variable, you can use the following code:

```
$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 distinct (unique) values from a column in the result set. It ensures that duplicate values are eliminated, and only unique values are returned.

The distinct() method is typically used in conjunction with the select() method to specify the columns from which to retrieve distinct values. For example:

```
$uniqueTitles = DB::table('posts')->select('title')->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.

Ans: To retrieve the first record from the "posts" table where the "id" is 2 using Laravel's query builder and store the result in the $posts variable, you can use the following code:

```
$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.

Ans: To retrieve the "description" column from the "posts" table where the "id" is 2 using Laravel's query builder and store the result in the $posts variable, you can use the following code:

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

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

Ans: The first() method and the find() method in Laravel's query builder are both used to retrieve single records from a table.

The first() method retrieves the first record that matches the query conditions. It returns a single instance of the query result as an object. For example:

```
$firstPost = DB::table('posts')->where('category', 'news')->first();
```

The find() method retrieves a record by its primary key. It expects the value of the primary key as a parameter and returns the corresponding record. For example:

```
$post = DB::table('posts')->find(1);
```

In this example, the find(1) method retrieves the record with id = 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.

Ans: To retrieve the "title" column from the "posts" table using Laravel's query builder and store the result in the $posts variable, you can use the following code:

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

Ans: To insert a new record into the "posts" table using Laravel's query builder, you can use the following code:

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

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: To update the "excerpt" and "description" columns of the record with the "id" of 2 in the "posts" table using Laravel's query builder, and set the new values to 'Laravel 10', you can use the following code:

```
$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.

Ans: To delete the record with the "id" of 3 from the "posts" table using Laravel's query builder, you can use the following code:

```
$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.

Ans: Aggregate methods in Laravel's query builder provide functionality to perform calculations on columns or retrieve statistical information from a set of records.

count(): Returns the number of records matching the query conditions. For example:

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

sum(): Calculates the sum of a specific column in the query result. For example:

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

avg(): Calculates the average (mean) value of a specific column in the query result. For example:

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

max(): Retrieves the maximum value of a specific column in the query result. For example:

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

min(): Retrieves the minimum value of a specific column in the query result. For example:

```
$minPrice = DB::table('products')->min('price');
```

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

Ans: The whereNot() method in Laravel's query builder is used to add a "not equal" condition to the query. It allows you to specify that a column should not match a certain value or an array of values.

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

```
$posts = DB::table('posts')->whereNot('category', 'news')->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?

Ans: The exists() and doesntExist() methods in Laravel's query builder are used to check the existence of records based on the query conditions.

exists(): Returns true if there is at least one record that matches the query conditions. For example:

```
$exists = DB::table('posts')->where('category', 'news')->exists();
```

doesntExist(): Returns true if there are no records that match the query conditions. For example:

```
$doesntExist = DB::table('posts')->where('category', 'news')->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.

Ans: To retrieve records from the "posts" table where the "min_to_read" column is between 1 and 5 using Laravel's query builder and store the result in the $posts variable, you can use the following code:

```
$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.

Ans: 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, you can use the following code:

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

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