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

Solution:

Laravel's query builder is a feature that allows developers to interact with databases in an easy and elegant way. It provides a set of methods and functions that enable you to build and execute database queries using a fluent, chainable syntax.
Rather than writing raw SQL statements, which can be complex and error-prone, the query builder lets you construct queries using simple and readable PHP code. It abstracts the underlying database engine and provides a consistent API for interacting with various database systems.
With the query builder, you can perform common database operations such as selecting data, inserting records, updating values, and deleting rows, among others. It offers a wide range of methods to handle complex query requirements, including filtering, sorting, joining tables, grouping, and aggregating data.
The query builder also helps with parameter binding, allowing you to securely pass user input or dynamic values into queries, guarding against SQL injection attacks. It automatically escapes input values, making your code more secure.
In short summary, Laravel's query builder simplifies database interactions by providing a fluent and expressive API, making it easier to read, write, and maintain database queries. It abstracts the complexities of SQL and offers a clean and concise way to work with databases, resulting in more efficient and manageable code.

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.

Solution:

```
$posts=DB::table('posts')->select('excerpt','description')->get();
```

3.Describe the purpose of the distinct() method in Laravel's query builder. How is it used in conjunction with the select() method?

Solution:

distinct() method is use to retrieve the unique data from database.

Here's an example of how the distinct() method can be used in conjunction with the select() method:

```
$uniqueEmails = DB::table('users') ->select('email') ->distinct() ->get();
```

In this example, the query retrieves only the distinct email addresses from the "users" table.

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.

Solution:

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

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.

Solution:

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

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

Solution:

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

```
$user = DB::table('users')->where('id', 1)->first();
```

In this example, the first() method is used to retrieve the first user record where the "id" column is equal to 1. It returns an object representing the first row of the result set. If no matching record is found, the first() method returns null.

```
$user = DB::table('users')->find(1);
```

In this example, the find(1) method is used to retrieve the user record with the primary key value of 1. It returns an object representing the found record.

If no record is found with the specified primary key value, the find() method returns null.

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.

Solution:

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

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

Solution:

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

```
        ]);
Print_r($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.

Solution:

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

Solution:

In Laravel's query builder, the aggregate methods count(), sum(), avg(), max(), and min() are used to perform calculations on a specific column of a table.

count(): This method is used to count the number of rows that match the query criteria. It is commonly used to get the total count of records in a table or the count of records that meet certain conditions.

sum(): This method is used to calculate the sum of a column's values. It is commonly used to get the total sum of numeric values in a column.

avg(): This method is used to calculate the average (mean) of a column's values. It is commonly used to get the average value of numeric data in a column.

max(): This method is used to retrieve the maximum value from a column. It is commonly used to find the highest value in a column.

min(): This method is used to retrieve the minimum value from a column. It is commonly used to find the lowest value in a column.

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

Solution:

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

In this example, the whereNot() method is used to retrieve records from the "users" table where the "status" column is not equal to "active". The whereNot() method takes two arguments: the column name and the value to compare against.

```
$excludedStatuses = ['inactive', 'suspended'];
$results = DB::table('users')
    ->whereNot('status', $excludedStatuses)
    ->get();
```

In this case, the whereNot() method is used to retrieve records where the "status" column is not equal to any value in the $excludedStatuses array. The query will exclude records with "inactive" and "suspended" statuses.

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?

Solution:

exists(): This method is used to check if any records exist in a table that match the given query criteria. It returns a boolean value indicating whether the query result contains any rows or not. If at least one record is found, the exists() method returns true; otherwise, it returns false.
doesntExist(): This method is the negation of the exists() method. It is used to check if no records exist in a table that match the given query criteria. It returns a boolean value indicating whether the query result is empty or not. If no records are found, the doesntExist() method returns true; otherwise, it returns false.

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.

Solution:

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

Solution:

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