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 that helps you work with databases in an easy and elegant way. It lets you create database queries without writing complex SQL statements.

Here are some key features and benefits of Laravel's query builder:

- This allows you to write database queries that are more concise and easier to understand.
- Laravel's query builder supports parameter binding, which helps prevent SQL injection attacks by automatically sanitizing user input.
- The query builder is designed to be database agnostic, meaning that you can write queries using its syntax regardless of the underlying database system.
- The query builder's fluent interface and expressive syntax contribute to improved code readability and maintainability.

Overall, Laravel's query builder offers a simple and elegant way to interact with databases. It abstracts the complexities of raw SQL and provides a consistent API for building queries. Whether you need to perform simple or complex database operations, the query builder empowers you to write efficient and readable code while leveraging the full potential of your chosen database system.

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.

```
$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?

In Laravel's query builder, the distinct() method is used to retrieve only unique records from a database query result.

here the example:

```
$users = DB::table('users')
    ->select('name', 'email')
    ->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.

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

return $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.

```
$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:

The first() method is used to retrieve the first record that matches the query conditions.

find() method:

The find() method is used to retrieve a record by 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.

```
$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.

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


return $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:

```
$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()----------

```
$totalUsers = DB::table('users')->count();
echo "Total users: " . $totalUsers;
```

---------- sum()----------

```
$totalSales = DB::table('orders')->sum('amount');
echo "Total sales: $" . $totalSales;
```

---------- avg()----------

```
$averageRating = DB::table('reviews')->avg('rating');
echo "Average rating: " . $averageRating;
```

---------- max()----------

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

echo "Highest price: $" . $highestPrice;
```

---------- min()----------

```
$lowestStock = DB::table('products')->min('stock');

echo "Lowest stock: " . $lowestStock;
```

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

Ans:

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

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 that match a specific condition.
- It returns a boolean value indicating whether any matching records exist or not.
- Example usage: DB::table('users')->where('status', 'active')->exists()

>>>doesntExist() method:

- The doesntExist() method is used to check if no records exist that match a specific condition.
- It returns a boolean value indicating whether no matching records exist or not.
- Example usage: 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.

```
$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.

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


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