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

Laravel's query builder is a powerful feature that allows developers to interact with databases using a fluent and intuitive syntax. It provides a simple and elegant way to build and execute database queries without the need for writing raw SQL statements.

With the query builder, you can perform various database operations such as selecting, inserting, updating, and deleting records. It abstracts the underlying database system and provides a consistent interface for working with different database engines, including MySQL, PostgreSQL, SQLite, and SQL Server.

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

1. Fluent syntax: The query builder uses a method chaining approach that allows you to construct queries in a readable and expressive manner. This makes it easier to write and understand complex queries.
2. Parameter binding: Laravel's query builder automatically handles parameter binding, which helps protect your application from SQL injection attacks. It allows you to specify placeholders in your queries and binds the values securely.
3. Database agnostic: The query builder provides a unified API for interacting with different database systems. It takes care of the subtle differences between database engines, allowing you to write database-agnostic code. This makes it easier to switch between databases without modifying your queries.
4. Query building methods: The query builder offers a wide range of methods to construct queries. You can chain methods like select, where, orderBy, join, groupBy, having, and more to build complex queries. These methods provide a high level of control over the generated SQL statements.

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.

```
use Illuminate\Support\Facades\DB;

$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?

The distinct() method in Laravel's query builder is used to return only unique rows from a database table. It is used in conjunction with the select() method to specify which columns should be returned.

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

```
use Illuminate\Support\Facades\DB;

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

if ($posts) {
        echo $posts->description;
} else {
        echo "No post found.";
}
```

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.

```
use Illuminate\Support\Facades\DB;

$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?

first() method:

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

- It returns a single instance of the model or a plain PHP object if no model is associated with the query.
- It does not require an argument and can be used directly after applying query conditions.
- The order in which the records are retrieved depends on the order specified in the query or the default order defined in the model.
- If no matching record is found, it returns null.

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

find() method:

- The find() method is used to retrieve a record by its primary key value.
- It requires the primary key value as an argument and returns the corresponding record if found.
- It is typically used when you know the specific primary key value you want to retrieve.
- If the record with the specified primary key does not exist, it returns null.

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

```
use Illuminate\Support\Facades\DB;

$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.

```
use Illuminate\Support\Facades\DB;

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

if ($inserted) {
        echo "Record inserted successfully!";
} else {
        echo "Failed to insert 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.

```
use Illuminate\Support\Facades\DB;
```

```
$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.

```
use Illuminate\Support\Facades\DB;

$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 the specified conditions.
- It returns the total count as an integer value.

```
use Illuminate\Support\Facades\DB;

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

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

sum():

- The sum() method is used to calculate the sum of a specific column's values that match the specified conditions.
- It returns the sum as a numeric value.

```
use Illuminate\Support\Facades\DB;

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

avg():

- The avg() method is used to calculate the average (mean) value of a specific column's values that match the specified conditions.
- It returns the average as a numeric value.

```
use Illuminate\Support\Facades\DB;

$averageRating = DB::table('reviews')->where('product_id',
1)->avg('rating');
```

max():

- The max() method is used to retrieve the maximum value of a specific column that matches the specified conditions.
- It returns the maximum value as a numeric or string value, depending on the column type.

```
use Illuminate\Support\Facades\DB;

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

min():

- The min() method is used to retrieve the minimum value of a specific column that matches the specified conditions.
- It returns the minimum value as a numeric or string value, depending on the column type.

```
use Illuminate\Support\Facades\DB;

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

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 that do not match a specific value or set of values in a given column. Here's how the whereNot() method is used and an example of its usage:

The whereNot() method is called on the query builder instance and takes two arguments: the column name and the value or an array of values that you want to exclude from the query results. It adds a "not equal" condition to the query, excluding the specified value(s) from the result set.

```
use Illuminate\Support\Facades\DB;

$users = DB::table('users')
        ->whereNot('role', 'admin')
        ->get();
```

In the example above, the whereNot() method is used to retrieve all users except those with the role of 'admin'. The whereNot('role', 'admin') condition ensures that the query excludes users whose role is equal to 'admin'. The get() method is then called to execute the query and retrieve the results.

```
$excludedRoles = ['admin', 'manager'];

$users = DB::table('users')
        ->whereNot('role', $excludedRoles)
        ->get();
```

In this example, the whereNot() method excludes users with the roles 'admin' and 'manager' from the query results.

The whereNot() method provides a convenient way to filter query results based on "not equal" conditions, allowing you to retrieve records that do not match specific values in a column.

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 the specified conditions.
- It returns a boolean value (true or false) indicating whether any matching records are found.
- It can be used with or without specifying conditions.
- If the query returns at least one record, the method will return true. Otherwise, it will return false.

Example usage:

```
use Illuminate\Support\Facades\DB;
```

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

if ($exists) {
        echo "At least one active user exists.";
} else {
        echo "No active users found.";
}
```

doesntExist() method:

- The doesntExist() method is used to check if no records exist that match the specified conditions.
- It returns a boolean value (true or false) indicating whether no matching records are found.
- It can be used with or without specifying conditions.
- If the query does not return any record, the method will return true. Otherwise, it will return false.

```
use Illuminate\Support\Facades\DB;

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

if ($doesntExist) {
        echo "No active users found.";
} else {
        echo "At least one active user 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.

```
use Illuminate\Support\Facades\DB;

$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.

```
use Illuminate\Support\Facades\DB;

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

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