Q1. Explain what Laravel's query builder is and how it provides a simple and elegant way to interact with databases.

Ans. In Laravel the database query builder provides an easy interface to create and run database queries. It can be used to perform all the database operations in our application, from basic DB Connection, CRUD, Aggregates, etc. and it works on all supported database systems like a champ.

The notable factor about query builder is that, since it uses the PHP Data Objects (PDO), we need not worry about SQL injection attacks (Make sure we don't inadvertently remove this protection). We can avoid all those lines of code to sanitize the data before feeding it to the DB.

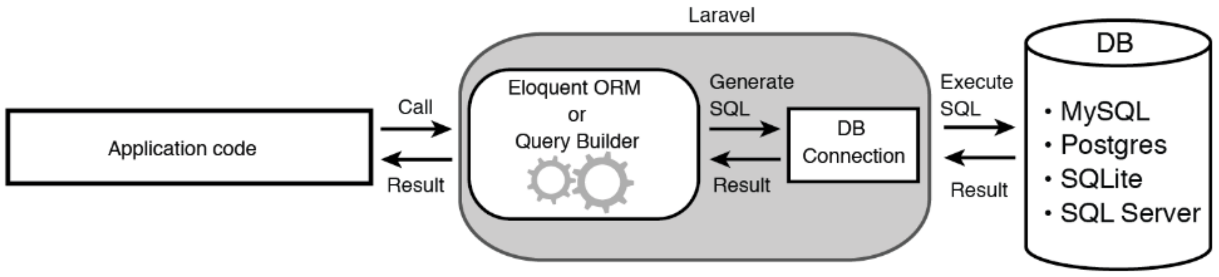

Figure 1.1 Database operations in Laravel

When the application's code contains any DB operations, behind the scenes Laravel converts this code into proper SQL statements that are then executed on any of the DB engines supported by Laravel. The result of the SQL execution is then returned all the way back to the application code and the application could use the result to either show data to the user, or process the data according to application's requirements(figure 1.1).

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. $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 or unique records from a table. It ensures that only unique values are returned in the result set, eliminating any duplicate rows. When used in conjunction with the select() method, the distinct() method applies the uniqueness constraint to the specified columns in the select() method. It ensures that only unique values are considered for those columns, and any duplicate values are removed from the result set. Here's an example to demonstrate the usage of distinct() and select() methods together:

$uniqueNames = DB::table('users')->select('name')->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.
```
$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.
```
$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. first() method:

The **first() method** retrieves the first record that matches the query conditions. It is typically used when you want to retrieve a single record based on specific conditions, such as filtering by a certain column value. It returns a single object instance representing the retrieved record.

```
$post = DB::table('posts')

        ->where('id', 2)

        ->first()
```

find() method:

The find() method retrieves a record by its primary key value. It is commonly used when you already know the primary key value of the record you want to retrieve. It returns a single object instance representing the retrieved record.

```
$post = DB::table('posts')->find(2);
```

Key difference:

The key difference between first() and find() is how they determine the record to retrieve. first() allows you to specify conditions in the query to filter records, while find() directly uses the primary key value to locate the record.

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.
```
$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.
```
$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.

```
Ans. $affectedRows = DB::table('posts')

                ->where('id', 2)

                ->update([

                    'excerpt' => 'Laravel 10',

                    'description' => 'Laravel 10'

                ]);



echo $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. $affectedRows = DB::table('posts')

                ->where('id', 3)

                ->delete();



echo $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 total count of records that match a specific condition. It returns the number of matching records.

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

sum(): The sum() method is used to calculate the sum of a specific column's values. It returns the total sum.

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

avg(): The avg() method is used to calculate the average of a specific column's values. It returns the average value.

$averageRating = DB::table('reviews')->avg('rating');

max(): The max() method is used to retrieve the maximum value from a specific column. It returns the highest value.

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

min(): The min() method is used to retrieve the minimum value from a specific column. It returns the lowest value.

$minStock = DB::table('items')->min('stock');

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

Ans. The whereNot and orWhereNot methods may be used to negate a given group of query constraints.

```
$users = DB::table('users')

          ->whereNot('status', 'inactive')

          ->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. exists() method:

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

$exists = DB::table('users')

```
            ->where('status', 'active')

            ->exists();
```

doesntExist() method:

The doesntExist() method is used to check if no records match a given query condition. It returns true if no records exist that satisfy the query condition, and false if there is at least one record that matches the condition.

```
$doesntExist = DB::table('users')

                ->where('status', 'deleted')

                ->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.

```
$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.

```
$affectedRows = DB::table('posts')

                ->where('id', 3)

                ->increment('min_to_read', 1);


echo $affectedRows
```