1. Laravel's query builder is a fluent interface for creating and executing database queries in PHP. Allows to interact with our database. Provides a wide range of methods. Highly flexible easy to use. Used to perform complex queries. Well-documented.

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

```
print_r($posts);
```

3. The distinct() method in Laravel's query builder is used to retrieve only unique values from a column. It removes duplicate values and ensures that each value appears only once in the output.

When used in conjunction with the select() method, the distinct() method is applied to the specified columns to retrieve unique values from those columns. It filters out duplicate values.

Example:

```
$categories = DB::table('products')->select('category')->distinct()->get();

    echo $categories;
```

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

```
echo $posts->description;
```

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

```
print_r($posts);
```

6. The first() and find() methods in Laravel's query builder are both used to retrieve a single record from the database.

first() returns the first record found in the database. If no matching model exist, it returns null.

find($id) takes an id and returns a single model. If no matching model exist, it returns null.

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

```
print_r($posts);
```

8. $result = DB::table('posts')->insert([

'title' => 'X',

'slug' => 'X',

'excerpt' => 'excerpt',

'description' => 'description',

'is_published' => true,

'min_to_read' => 2

]);

print_r($result);

9. $affectedRows = DB::table('posts')->where('id', 2)->update([

'excerpt' => 'Laravel 10',

'description' => 'Laravel 10'

]);

print_r($affectedRows);

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

print_r($affectedRows);

11.The aggregate methods in Laravel's query builder used to perform calculations on a specific column or set of columns in a database table. These methods allow you to retrieve aggregated data based on specific criteria.

count:

The count method is used to retrieve the number of records that match a specific condition or the total number of records in a table.

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

echo $count;

sum:

The sum method is used to calculate the sum of a column's values.

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

avg:

The avg method is used to calculate the average value of a column.

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

max:

The max method is used to retrieve the maximum value of a column.

```
$highestScore = DB::table('scores')->max('score');
echo $highestScore;
```

min:

The min method is used to retrieve the minimum value of a column.

```
$lowestTemperature = DB::table('temperatures')->min('temperature');
echo $lowestTemperature;
```

12. The whereNot method is used to add a "not equal" condition to the query. It allows to retrieve records where a specific column's value is not equal to a given value or an array of values.

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

13. exists:

The exists method is used to check if there are any records that match a specific condition.

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

```
echo $exists;
```

doesntExist:

The doesntExist method is used to check if there are no records that match a specific condition.

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

echo $doesntExist;
```

14.
```
$posts = DB::table('posts')->whereBetween('min_to_read', [1, 5])->get();

print_r($posts);
```

15.
```
$affectedRows = DB::table('posts')->where('id', 3)->increment('min_to_read');

print_r($affectedRows);
```