Assignment-17

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

Answer : A key component of the Laravel framework that gives developers an easy and attractive method to deal with databases is the query builder. It makes database operations more legible and effective by enabling you to construct and execute database queries using a fluid, chainable interface.

You may create queries that are independent of the underlying database engine thanks to the query builder. It is extremely adaptable and supports a wide range of database systems, including MySQL, PostgreSQL, SQLite, and SQL Server.

You may choose, insert, update, and delete records among other typical database actions using the query builder. It provides a variety of query construction techniques, such as conditions, joins, aggregations, sorting, and result limiting. With this flexibility, you may create sophisticated queries that are yet readable and straightforward.

The query builder's capacity to defend against SQL injection attacks by automatically sanitizing user input is one of its primary benefits. Your program will be more secure because it manages parameter binding and makes sure that user-supplied data is correctly escaped.

Furthermore, you can combine the power of both functionalities thanks to the query builder's smooth integration with Laravel's Eloquent ORM (Object-Relational Mapping) framework. Depending on the complexity of the data operations in your application, you may quickly switch between utilizing the query builder and working with Eloquent models.

By offering a clear and simple interface, Laravel's query builder streamlines and simplifies database interactions. It promotes code readability, improves security, and enables compatibility with numerous database systems while abstracting the complexity of raw SQL queries. It is favored because of these features.

Question 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.

First, I ran migration with two table called post and column and then I give some fake value inside those two table by DatabaseSeeder and PostFactory.

Here is some screenShoots

```
2023_06_11_171535_create_posts_table.php
2023_06_11_173507_create_users_table.php
```

FactorySeeder :

```php
public function definition(): array
{

    $title = fake()->sentence;
    $slug = Str::slug( title: $title, separator: '-' );
    return [
        'user_id'=>rand( min: 1, max: 5),
        'title'        => $title,
        'slug'         => $slug,
        'excerpt'      => fake()->sentence,
        'description'  => fake()->sentence,
        'is_published' => fake()->boolean,
        'min_to_read'  => fake()->randomDigit(),
    ];
}
```

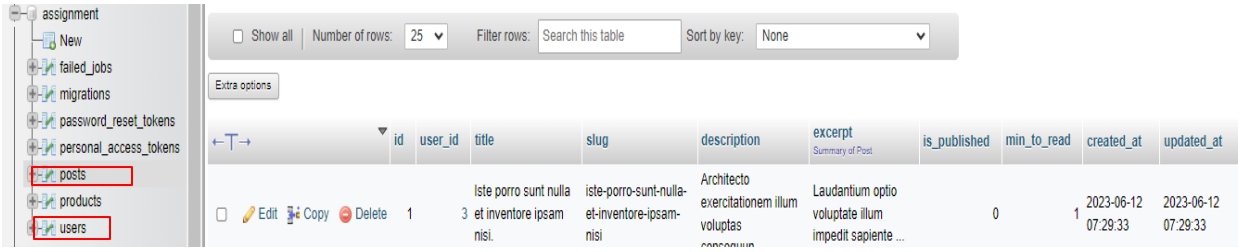
DatabaseSeeder :

```php
0 references | 0 implementations
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    0 references | 0 overrides
    public function run(): void
    {
        // \App\Models\User::factory(10)->create();

        // \App\Models\User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);

        Post::factory( count: 20)->create();
    }
}
```

Then I made a database and retrieve "excerpt" and "description" from posts table



I have create a postController –resource where I found index method. I have also made a web route where I made a Route for posts url. When I hit posts route it have come with this "excerpt" and "description" from posts table

PostController :

```
class PostsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    0 references | 0 overrides
    public function index()
    {
        $posts= DB::table( table: 'posts')->select( columns: 'excerpt','description')->get();
        return $posts;
    }
```

Web Route :

```
Route::resource( name: '/posts', controller: PostsController::class);
```

Output :

127.0.0.1:8000/posts


```
[
    {
        "excerpt": "Laudantium optio voluptate illum impedit sapiente optio eveniet.",
        "description": "Architecto exercitationem illum voluptas consequuntur quas quidem ducimus."
    },
    {
        "excerpt": "Vitae soluta harum eum tempore iste.",
        "description": "Est quisquam quos quis suscipit commodi reprehenderit numquam deleniti."
    },
    {
        "excerpt": "Ab qui nulla eveniet id assumenda.",
        "description": "Consequatur tempore ut vel ut animi sunt qui."
    },
    {
        "excerpt": "Maxime suscipit et temporibus provident tempore.",
        "description": "Consequatur nemo earum provident iste consectetur eius ut."
    },
    {
        "excerpt": "Velit et itaque nemo neque.",
        "description": "Fuga deleniti rerum odit eum tempore."
    },
    {
        "excerpt": "Voluptas vel soluta dolorem ea voluptas quia ut.",
        "description": "Aspernatur omnis quaerat eius rerum minus aliquam ratione."
    },
    {
        "excerpt": "Quo nisi beatae alias numquam molestiae necessitatibus.",
        "description": "Ipsam a sunt quia reiciendis enim id et."
    },
    {
        "excerpt": "Nesciunt necessitatibus culpa fuga et esse blanditiis.",
        "description": "Libero veniam ut sunt enim dolor tenetur."
    },
    {
        "excerpt": "Odio nesciunt debitis et eos ut.",
        "description": "Non autem officia ducimus beatae ea sint."
    },
    {
        "excerpt": "Ducimus consequuntur quis dolor reiciendis vitae aliquid.",
        "description": "Ullam doloremque quia possimus magnam voluptas."
    },
    {
        "excerpt": "Minima pariatur et doloribus vel mollitia.",
        "description": "Quaerat et eum enim non et ea enim."
    },
    {
        "excerpt": "Sit tenetur atque necessitatibus rerum.",
        "description": "Eum ullam consequatur debitis sed sunt."
    },
```

Question 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 the Laravel query builder is used to retrieve a query result set with distinct values for a given column or set of columns. It does this by removing duplicate values, ensuring that only unique records are returned.

When combined with the select() function, the distinct() method changes the query so that it only returns distinct values from the specified columns. This is helpful if you want to obtain a certain set of records based on a set of columns while avoiding duplicate values in those columns.

Here's an example to illustrate the usage of distinct() with select():

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

Question 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.

<u>Here are the screenshoots :</u>

```
$posts=DB::table( table: 'posts')->where( column: 'id', operator: 2)->first();
dd( vars[0]: $posts->description);
```

Output from the Data Table:

←T→		id	user_id	title	slug	description	excerpt Summary of Post	is_published	min_to_read	created_at	updated_at
Edit Copy Delete		1	3	Iste porro sunt nulla et inventore ipsam nisi.	iste-porro-sunt-nulla-et-inventore-ipsam-nisi	Architecto exercitationem illum voluptas consequun...	Laudantium optio voluptate illum impedit sapiente ...	0	1	2023-06-12 07:29:33	2023-06-12 07:29:33
Edit Copy Delete		2	5	A deleniti rerum sint eum.	a-deleniti-rerum-sint-eum	Est quisquam quos quis suscipit commodi reprehende...	Vitae soluta harum eum tempore iste.	1	3	2023-06-12 07:29:34	2023-06-12 07:29:34

<u>Output in browser:</u>

```
"Est quisquam quos quis suscipit commodi reprehenderit numquam deleniti." // app\Http\Controllers\PostsController.php:24
```

Question 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.

<u>Here are the screenshoots :</u>

```
$posts=DB::table( table: 'posts')->where( column: 'id', operator: 2)->value( column: 'descrip
dd( vars[0]: $posts);
```

<u>Output in browser:</u>

```
"Est quisquam quos quis suscipit commodi reprehenderit numquam deleniti." // app\Http\Controllers\PostsController.php:27
```

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

For retrieving a single record from the database in Laravel's query builder, the first() and find() methods are both used. To specify the record to be retrieved, they each have a unique method.

first() retrieves the first record based on query criteria (in this case, the name), while **find()** retrieves a record based on its primary key value (in this case, the ID).

Here are the example bellow :

- ⇨ **first()** :

```
$posts=DB::table( table: 'posts')->where( column: 'id', operator: 2)->first();
dd( vars[0]: $posts);
```

Output :

```
{#291 ▼ // app\Http\Controllers\PostsController.php:24
  +"id": 2
  +"user_id": 5
  +"title": "A deleniti rerum sint eum."
  +"slug": "a-deleniti-rerum-sint-eum"
  +"description": "Est quisquam quos quis suscipit commodi reprehenderit numquam deleniti."
  +"excerpt": "Vitae soluta harum eum tempore iste."
  +"is_published": 1
  +"min_to_read": 3
  +"created_at": "2023-06-12 07:29:34"
  +"updated_at": "2023-06-12 07:29:34"
}
```

⇨ **find() :**

```
32      $posts=DB::table( table: 'posts')->find( id: 2);
33      dd( vars[0]: $posts);
```

Output :

```
{#291 ▼ // app\Http\Controllers\PostsController.php:33
  +"id": 2
  +"user_id": 5
  +"title": "A deleniti rerum sint eum."
  +"slug": "a-deleniti-rerum-sint-eum"
  +"description": "Est quisquam quos quis suscipit commodi reprehenderit numquam deleniti."
  +"excerpt": "Vitae soluta harum eum tempore iste."
  +"is_published": 1
  +"min_to_read": 3
  +"created_at": "2023-06-12 07:29:34"
  +"updated_at": "2023-06-12 07:29:34"
}
```

Question 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.

⇨ List of column values to be retrieved (pluck method): The pluck() function is used to extract a single column value from a query's first result.

Here are the screenshoots :

```
35      $posts=DB::table( table: 'posts')->pluck( column: 'title');
36      dd( vars[0]: $posts);
37
```

Output :

```
Illuminate\Support\Collection {#289 ▼ // app\Http\Controllers\PostsController.php:36
  #items: array:20 [▼
    0 => "A deleniti rerum sint eum."
    1 => "Ab unde quasi ex sunt et."
    2 => "Accusamus aperiam maiores atque sapiente voluptas."
    3 => "Aut quis laboriosam voluptate eveniet tenetur."
    4 => "Autem est et eaque et illum omnis."
    5 => "Consequatur dignissimos aspernatur voluptatem quis aut."
    6 => "Consequatur magnam est accusantium inventore."
    7 => "Dolorem incidunt aut tenetur asperiores quas quam fugit sit."
    8 => "Facilis exercitationem ipsa aperiam."
    9 => "Iste porro sunt nulla et inventore ipsam nisi."
    10 => "Minus voluptatum itaque sed modi sint deserunt eum maxime."
    11 => "Molestiae et qui eum culpa nihil incidunt soluta eaque."
    12 => "Nam aspernatur commodi fugit maxime quod deleniti repellat animi."
    13 => "Officia dolorem rerum odio hic magni."
    14 => "Quasi rem eos dolorum nemo explicabo dolorem."
    15 => "Ratione illo aut aperiam ut a aspernatur excepturi."
    16 => "Recusandae nobis libero omnis cumque."
    17 => "Voluptas et molestias ea labore nulla nostrum."
    18 => "Voluptas qui magni veniam sint explicabo eos non."
    19 => "Voluptatem sed earum a soluta consectetur aut voluptatibus."
  ]
  #escapeWhenCastingToString: false
}
```

Question 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.

Here are the ScreenShoots :

Insert :

```php
    public function up(): void
    {
        Schema::create( table: 'posts', callback: function (Blueprint $table) {
            $table->id();
            $table->foreignId( column: 'user_id' )
            ->constrained( table: 'users' )
            ->cascadeOnDelete();
        $table->string( column: 'title' )
            ->unique();
        $table->string( column: 'slug' )
            ->unique();
        $table->longText( column: 'description' );
        $table->text( column: 'excerpt' )
            ->comment( comment: 'Summary of Post' );
        $table->boolean( column: 'is_published' )
            ->default( value: false );
        $table->integer( column: 'min_to_read' )
            ->nullable();
        $table->timestamps();
        });
    }
```

Output :

Server: 127.0.0.1 » Database: assignment » Table: posts

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Table structure | Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
☐	2	user_id	bigint(20)		UNSIGNED	No	None			Change Drop More
☐	3	title	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
☐	4	slug	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
☐	5	description	longtext	utf8mb4_unicode_ci		No	None			Change Drop More
☐	6	excerpt	text	utf8mb4_unicode_ci		No	None	Summary of Post		Change Drop More
☐	7	is_published	tinyint(1)			No	0			Change Drop More
☐	8	min_to_read	int(11)			Yes	NULL			Change Drop More
☐	9	created_at	timestamp			Yes	NULL			Change Drop More
☐	10	updated_at	timestamp			Yes	NULL			Change Drop More

Question 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.

Here are the screenshoots bellow :

```
38
39      $updated=DB::table( table: 'posts' )
40      ->where( column: 'id', operator: 2 )
41     ->update( values: [
42          'excerpt'     => 'Laravel 10',
43          'description' => 'Laravel 10',
44     ] );
45
46      dd( vars[0]: $updated);
47
```

Output :

Extra options

←T→	id	user_id	title	slug	description	excerpt Summary of Post	is_published	min_to_read	created_at	updated_at
Edit Copy Delete	1	3	Iste porro sunt nulla et inventore ipsam nisi.	iste-porro-sunt-nulla-et-inventore-ipsam-nisi	Architecto exercitationem illum voluptas consequun...	Laudantium optio voluptate illum impedit sapiente ...	0	1	2023-06-12 07:29:33	2023-06-12 07:29:33
Edit Copy Delete	2	5	A deleniti rerum sint eum.	a-deleniti-rerum-sint-eum	Laravel 10	Laravel 10	1	3	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	3	4	Aut quis laboriosam voluptate eveniet tenetur.	aut-quis-laboriosam-voluptate-eveniet-tenetur	Consequatur tempore ut vel ut animi sunt qui.	Ab qui nulla eveniet id assumenda.	0	9	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	4	2	Facilis exercitationem ipsa aperiam.	facilis-exercitationem-ipsa-aperiam	Consequatur nemo earum provident iste consectetur ...	Maxime suscipit et temporibus provident tempore.	0	2	2023-06-12 07:29:34	2023-06-12 07:29:34

Question 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.

Here are the screenShoots :

Before delete column 3 :

Server: 127.0.0.1 » Database: assignment » Table: posts

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

	id	user_id	title	slug	description	excerpt Summary of Post	is_published	min_to_read	created_at	updated_at
Edit Copy Delete	1	3	Iste porro sunt nulla et inventore ipsam nisi.	iste-porro-sunt-nulla-et-inventore-ipsam-nisi	Architecto exercitationem illum voluptas consequun...	Laudantium optio voluptate illum impedit sapiente ...	0	1	2023-06-12 07:29:33	2023-06-12 07:29:33
Edit Copy Delete	2	5	A deleniti rerum sint eum.	a-deleniti-rerum-sint-eum	Laravel 10	Laravel 10	1	3	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	3	4	Aut quis laboriosam voluptate eveniet tenetur.	aut-quis-laboriosam-voluptate-eveniet-tenetur	Consequatur tempore ut vel ut animi sunt qui.	Ab qui nulla eveniet id assumenda.	0	9	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	4	2	Facilis exercitationem ipsa aperiam.	facilis-exercitationem-ipsa-aperiam	Consequatur nemo earum provident iste consectetur ...	Maxime suscipit et temporibus provident tempore.	0	2	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	5	1	Recusandae nobis libero omnis cumque.	recusandae-nobis-libero-omnis-cumque	Fuga deleniti rerum odit eum tempore.	Velit et itaque nemo neque.	0	4	2023-06-12 07:29:34	2023-06-12 07:29:34

After I use this :

```
48
49    DB::table( table: 'posts')
50    ->where( column: 'id',  operator: 3)
51    ->delete();
52
```

Column number 3 deleted from the posts Table

Server: 127.0.0.1 » Database: assignment » Table: posts

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

	id	user_id	title	slug	description	excerpt Summary of Post	is_published	min_to_read	created_at	updated_at
Edit Copy Delete	1	3	Iste porro sunt nulla et inventore ipsam nisi.	iste-porro-sunt-nulla-et-inventore-ipsam-nisi	Architecto exercitationem illum voluptas consequun...	Laudantium optio voluptate illum impedit sapiente ...	0	1	2023-06-12 07:29:33	2023-06-12 07:29:33
Edit Copy Delete	2	5	A deleniti rerum sint eum.	a-deleniti-rerum-sint-eum	Laravel 10	Laravel 10	1	3	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	4	2	Facilis exercitationem ipsa aperiam.	facilis-exercitationem-ipsa-aperiam	Consequatur nemo earum provident iste consectetur ...	Maxime suscipit et temporibus provident tempore.	0	2	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	5	1	Recusandae nobis libero omnis cumque.	recusandae-nobis-libero-omnis-cumque	Fuga deleniti rerum odit eum tempore.	Velit et itaque nemo neque.	0	4	2023-06-12 07:29:34	2023-06-12 07:29:34
Edit Copy Delete	6	3	Nam aspernatur commodi fugit maxime quod deleniti ...	nam-aspernatur-commodi-fugit-maxime-quod-deleniti-...	Aspernatur omnis quaerat eius rerum minus aliquam ...	Voluptas vel soluta dolorem ea voluptas quia ut.	0	6	2023-06-12 07:29:34	2023-06-12 07:29:34

Question 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.

Here are the details :

- **Count() : The count() method is commonly used to determine the number of records in a table or the count of specific rows that meet certain criteria.**

Example :

```
DB::table( table: 'posts')->count();
```

- **Sum() : sum() is helpful when you want to calculate the total value of a numeric column, such as the sum of sales amounts or the sum of quantities.**

Example :

```
DB::table( table: 'posts')->sum( column: 'min_to_read');
```

- **Average() : With avg(), you can easily find the average value of a column, which is useful for obtaining the average rating of products or the average age of users.**

Example :

```
DB::table( table: 'posts')->avg( column: 'min_to_read');
```

- **Max() : When you need to find the highest value in a column, the max() method comes in handy, allowing you to identify the maximum price in a list of products or the highest score in a game.**

Example :

```
DB::table( table: 'posts')->where( column: 'is_published',  operator: true)->max( column: 'min_to_read');
```

- **Min() : Conversely, the min() method helps you find the lowest value in a column, such as the minimum temperature in a dataset or the lowest price of a product.**

Example :

```
DB::table( table: 'posts')->where( column: 'is_published',  operator: true)->min( column: 'min_to_read');
```

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

<u>Explain :</u> You may include a "not equal" condition in your database query by using the whereNot() method in Laravel's query builder. Records are removed from the result set if they don't meet the stated requirement.

<u>Example :</u>

```
DB::table('posts')->whereNot('min_to_read', '>', 1)->get();
```

Question 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?

Here are the explanation bellow-

- **exists(): The exists() method determines whether at least one record in the provided table exists and meets the supplied criteria. If a matching record is found, it produces a boolean value of true; otherwise, it returns a value of false. When you want to make that a record exists before taking specific actions, this approach can be helpful.**

Example :

```
69
70      if(DB::table( table: 'posts')->where( column: 'id',  operator: 343543)->exists()) {
71          echo 'This is exist';
72      } else {
73          echo 'This does not exist!';
74      }
75
```

Output :

This does not exist!

- **doesntExist() : The doesntExist() method, on the other hand, is the opposite of the exists() method. It looks to see whether there are any records in the provided table that satisfy the given criteria. In the absence of a matching record, it returns true; otherwise, it returns false. This technique is useful when you want to make sure a record doesn't already exist before you take specific actions, such making a new record to prevent duplications.**

Example :

```php
if(DB::table( table: 'posts')->where( column: 'id',  operator: 343543)->doesntExist()) {
    echo 'Yes! This is match the post';
} else {
    echo 'Ahh, I have not found a match';
}
```

Question 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.

Here are the details:

Method :

```php
$posts = DB::table( table: 'posts')->whereBetween( column: 'min_to_read',  values: [1, 5])->get();
dd( vars[0]: $posts);
```

Output :

```
Illuminate\Support\Collection {#292 ▼ // app\Http\Controllers\PostsController.php:85
  #items: array:14 [▼
    0 => {#290 ▼
      +"id": 1
      +"user_id": 3
      +"title": "Iste porro sunt nulla et inventore ipsam nisi."
      +"slug": "iste-porro-sunt-nulla-et-inventore-ipsam-nisi"
      +"description": "Architecto exercitationem illum voluptas consequuntur quas quidem ducimus."
      +"excerpt": "Laudantium optio voluptate illum impedit sapiente optio eveniet."
      +"is_published": 0
      +"min_to_read": 1
      +"created_at": "2023-06-12 07:29:33"
      +"updated_at": "2023-06-12 07:29:33"
    }
    1 => {#295 ▼
      +"id": 2
      +"user_id": 5
      +"title": "A deleniti rerum sint eum."
      +"slug": "a-deleniti-rerum-sint-eum"
      +"description": "Laravel 10"
      +"excerpt": "Laravel 10"
      +"is_published": 1
      +"min_to_read": 3
      +"created_at": "2023-06-12 07:29:34"
      +"updated_at": "2023-06-12 07:29:34"
    }
    2 => {#293 ▼
      +"id": 4
      +"user_id": 2
      +"title": "Facilis exercitationem ipsa aperiam."
      +"slug": "facilis-exercitationem-ipsa-aperiam"
      +"description": "Consequatur nemo earum provident iste consectetur eius ut."
      +"excerpt": "Maxime suscipit et temporibus provident tempore."
      +"is_published": 0
      +"min_to_read": 2
      +"created_at": "2023-06-12 07:29:34"
      +"updated_at": "2023-06-12 07:29:34"
    }
    3 => {#294 ▶}
    4 => {#296 ▶}
    5 => {#297 ▶}
    6 => {#298 ▶}
    7 => {#299 ▶}
    8 => {#300 ▶}
    9 => {#301 ▶}
```

Question 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.

Here are the details :

Firstly, I need to say that I have no column number 3 inside my database because I deleted it before. So, alternatively I have used column number 4 using without 3.

Required column :

```
$posts = DB::table( table: 'posts')->where( column: 'id', operator: 3)->increment( column: 'min_to_read',  amount: 1);
dd( vars[0]: $posts);
```

Alternative Column Number mention 4:

```
$posts = DB::table( table: 'posts')->where( column: 'id', operator: 4)->increment( column: 'min_to_read',  amount: 1);
dd( vars[0]: $posts);
```

Here it comes :

```
1 // app\Http\Controllers\PostsController.php:88
```

Here is a simple database diagram screenshot where i wanted to show the relation between “posts” and “user” table