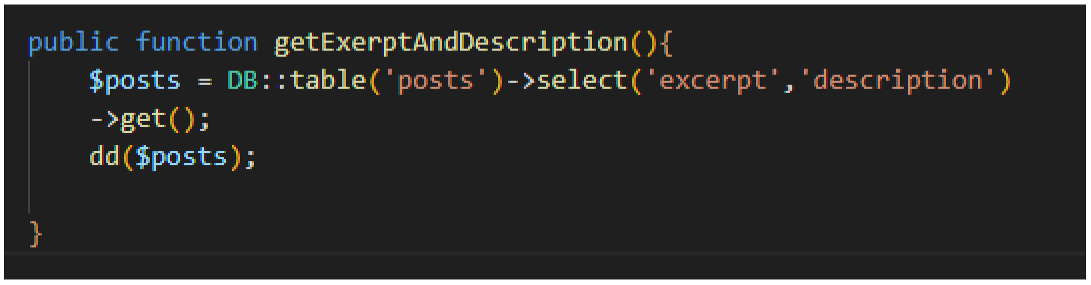
Assignment-17

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

Answer: Laravel's query builder is a feature that provides a simple and elegant way to interact with databases in the Laravel framework. It allows developers to build database queries using a fluent, chainable interface. The query builder abstracts the database system, offers methods for constructing queries, handles parameter binding, and integrates with the Eloquent ORM. It is database agnostic and supports various engines. The query builder improves code readability, enhances security, and provides query logging and debugging capabilities. Overall, it streamlines the database interaction process and makes development more efficient.

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.

Answer:



```
public function getExerptAndDescription(){
    $posts = DB::table('posts')->select('excerpt','description')
    ->get();
    dd($posts);

}
```

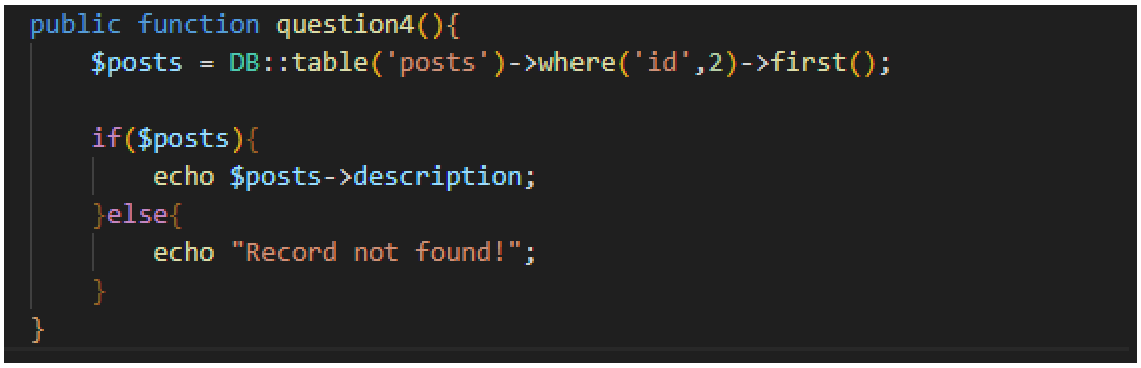
3.Describe the purpose of the distinct() method in Laravel's query builder. How is it used in conjunction with the select() method?

Answer: The **distinct()** method is used to retrieve unique values from a column or combination of columns in the query result.

the **distinct()** method in conjunction with the **select()** method, allows to specify the columns and retrieve only unique values or combinations. We can select single or multiple columns in **select()** to find out the unique values based on the given columns. It enables to filter out duplicates and obtain distinct results directly from the query.

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.

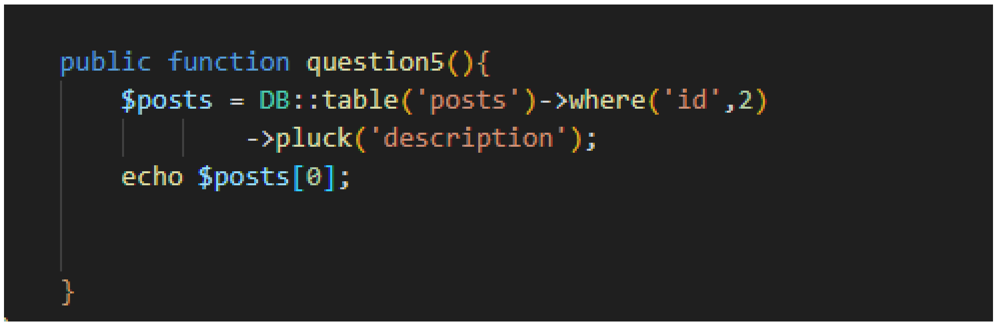
Answer:



```php
public function question4(){
    $posts = DB::table('posts')->where('id',2)->first();

    if($posts){
        echo $posts->description;
    }else{
        echo "Record not 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.

Answer:



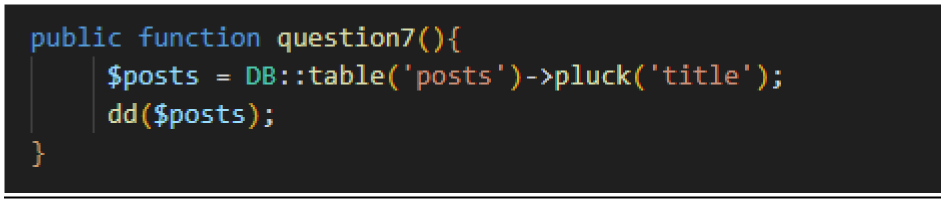
```php
public function question5(){
    $posts = DB::table('posts')->where('id',2)
            ->pluck('description');
    echo $posts[0];


}
```

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

Answer: The **first()** method fetches the first record that matches the query criteria while the **find()** method fetches a record by its primary key. Both methods are used to retrieves single record but **first()** method works based on **query condition**, while **find()** methods works based on **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.

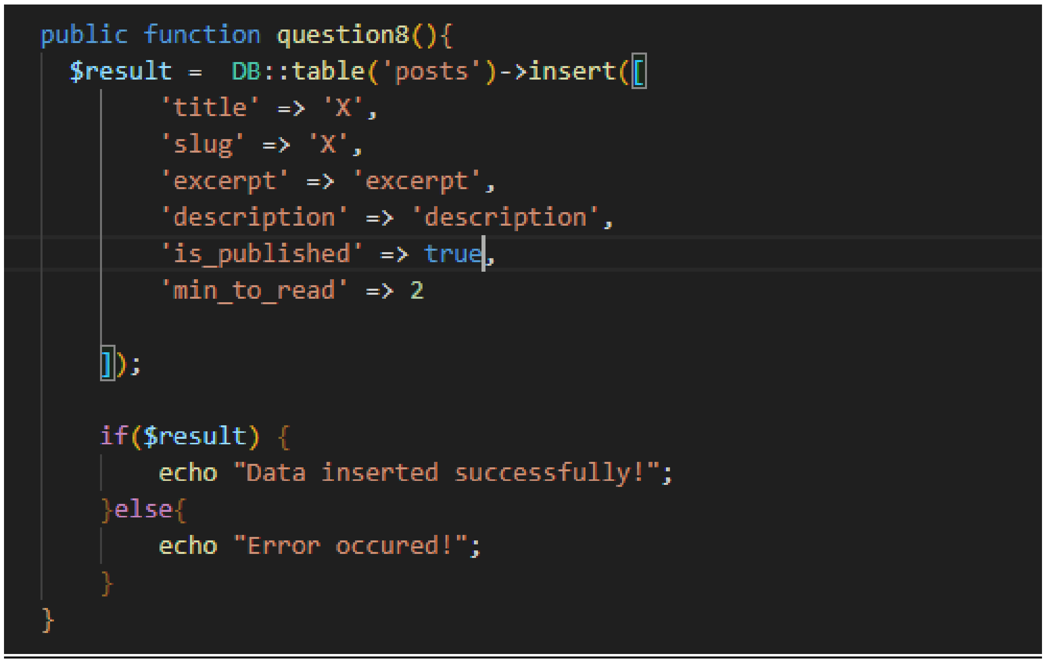
Answer:



```php
public function question7(){
        $posts = DB::table('posts')->pluck('title');
        dd($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.

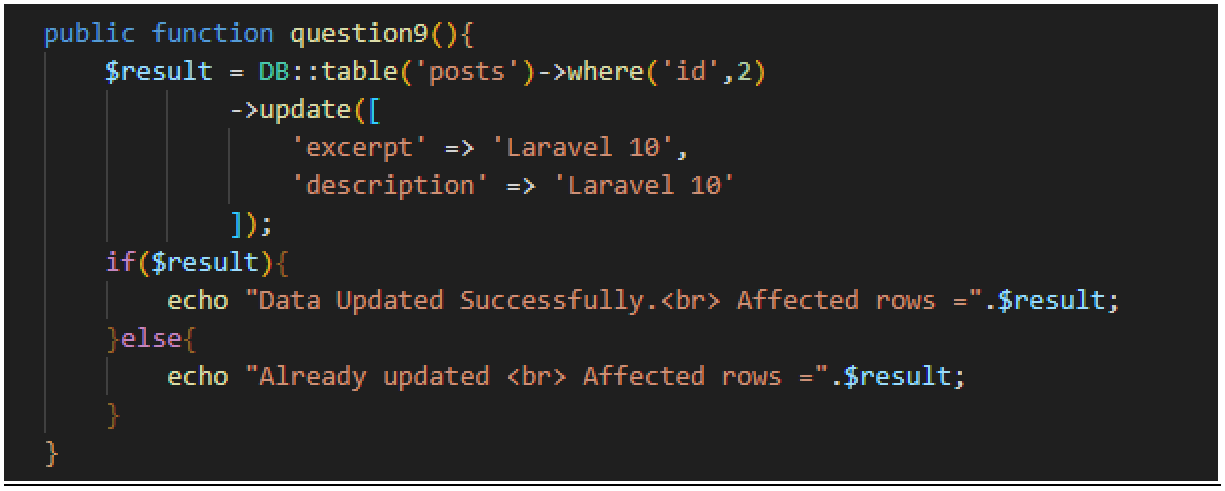
Answer:



```php
public function question8(){
  $result =  DB::table('posts')->insert([
        'title' => 'X',
        'slug' => 'X',
        'excerpt' => 'excerpt',
        'description' => 'description',
        'is_published' => true,
        'min_to_read' => 2

    ]);

    if($result) {
        echo "Data inserted successfully!";
    }else{
        echo "Error occured!";
    }
}
```

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.

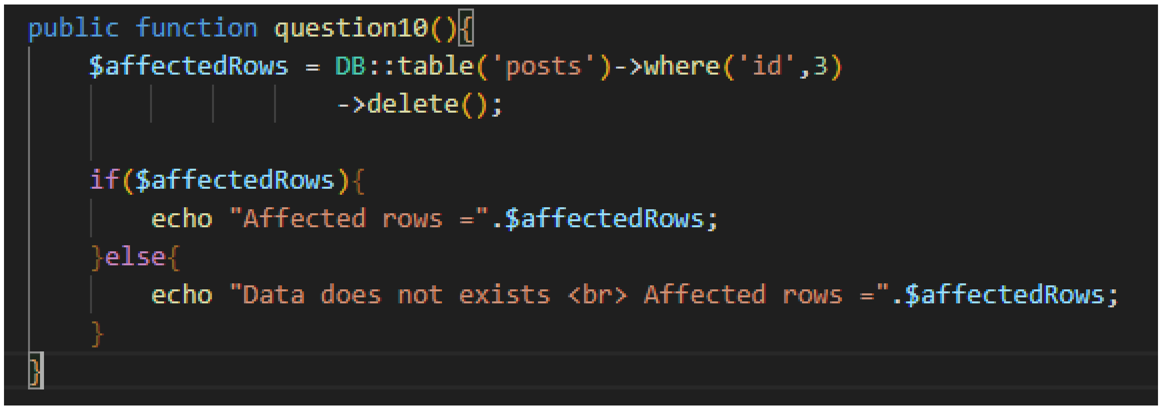
Answer:



```php
public function question9(){
    $result = DB::table('posts')->where('id',2)
            ->update([
                'excerpt' => 'Laravel 10',
                'description' => 'Laravel 10'
            ]);
    if($result){
        echo "Data Updated Successfully.<br> Affected rows =".$result;
    }else{
        echo "Already updated <br> Affected rows =".$result;
    }
}
```

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.

Answer:



```
public function question10(){
    $affectedRows = DB::table('posts')->where('id',3)
                ->delete();

    if($affectedRows){
        echo "Affected rows =".$affectedRows;
    }else{
        echo "Data does not exists <br> 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.

Answer:

The aggregate methods in laravel query builder are used to perform calculations on the values of specific columns in a table. Aggregate methods are used to generate reports, calculating statistics and many other kinds of calculation purposed works. There are several aggregate methods in Laravel query builders these are:

- **count()** : The count() method returns the total number of rows in a table based on the query criteria.

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

  In this example count() method will return the total number of rows in posts table.

- **sum()** : The sum() method add the total values of a specific column in a table and returns the sum.

  $totalReadTime = DB::table('posts)->sum('min_to_read);

  In this example sum method will return the total read time of all the posts in posts table.

- **avg():** The avg() method calculates and returns the values from a specific column in a table.

  $avgReadTime = DB::table('posts')->avg('min_to_read');

  In this example avg() method will return the total read time for all of the posts.

- **max() :** The max() method calculates and returns the biggest value from a specific column in a table.

$maxReadTime = DB::table('posts')->max('min_to_read');

Here max() method will return the maximum values from min_to_read column in posts table.

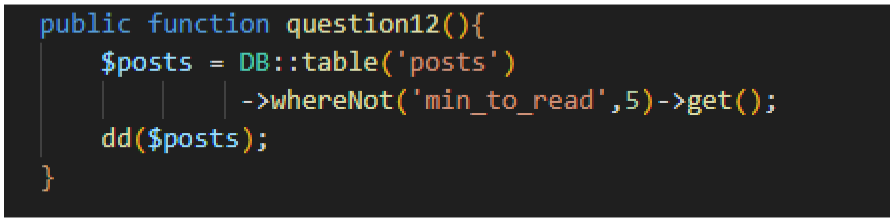
- **min() :** The min() method calculates and returns the minimum value from a specific column in a table.

  $minReadTime = DB::table('posts')->min('min_to_read');

  Here min() method will return the minimum values from min_to_read column in posts table

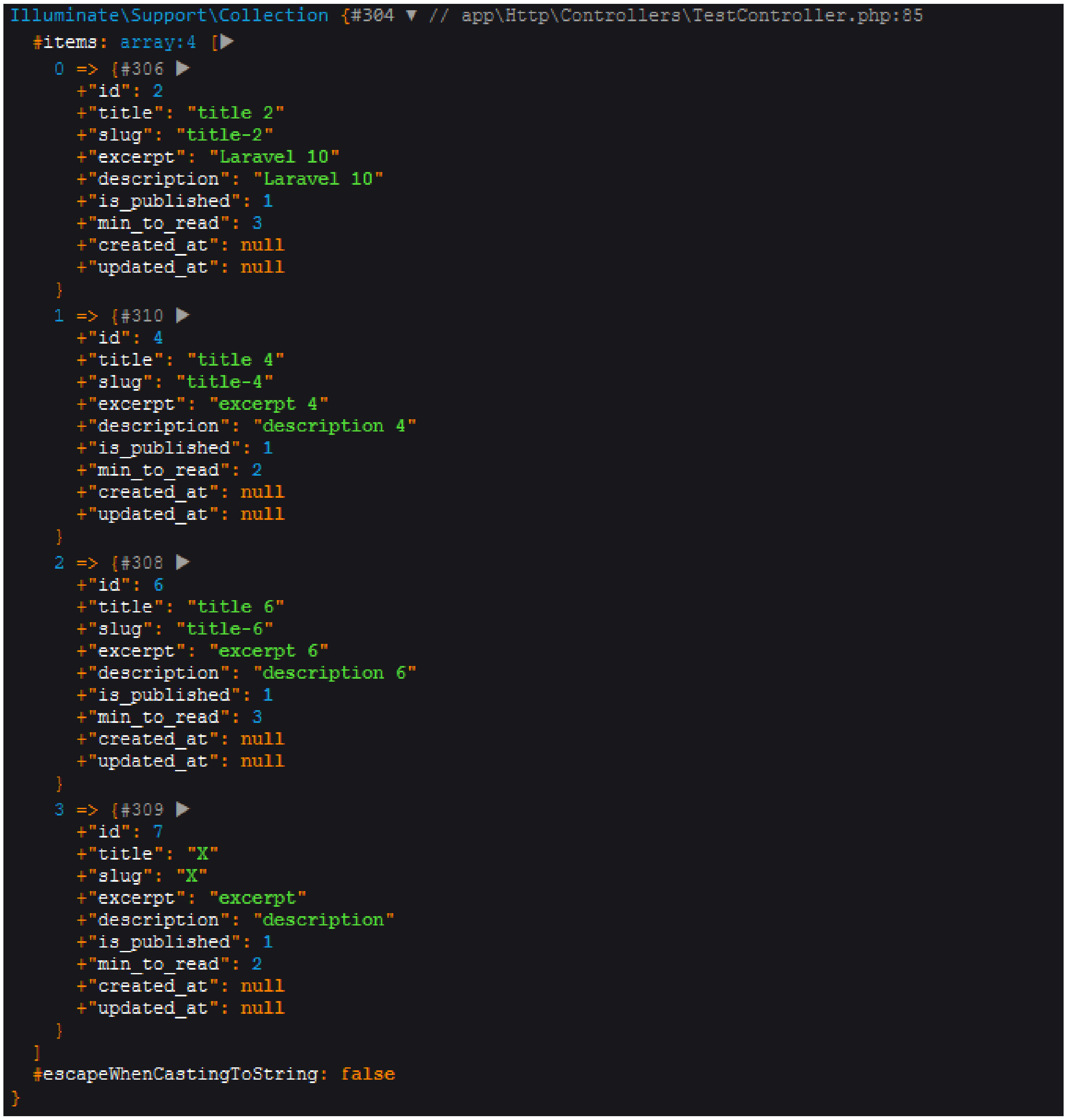
12.Describe how the whereNot() method is used in Laravel's query builder. Provide an example of its usage.

Answer: The **whereNot()** method in Laravel's query builder is used to add a "not equal" condition to a query. It allows to retrieve records where a specific column's value is not equal to a given value or does not match a set of values. It is useful for excluding specific values or a set of values from the query results.



```
public function question12(){
    $posts = DB::table('posts')
                ->whereNot('min_to_read',5)->get();
    dd($posts);
}
```

Here is a example code snippet of whereNot() method. Here the code will return all records excepts which records values of min_to_read value is 5.

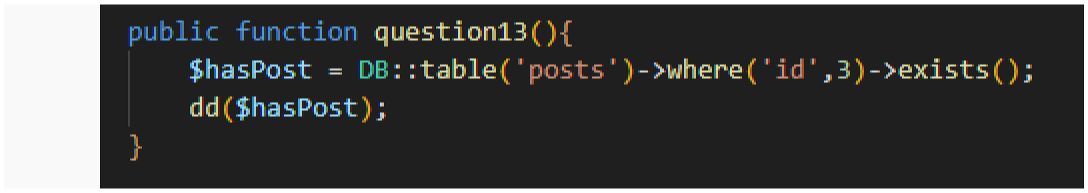


```
Illuminate\Support\Collection {#304 ▼ // app\Http\Controllers\TestController.php:85
  #items: array:4 [▶
    0 => {#306 ▶
      +"id": 2
      +"title": "title 2"
      +"slug": "title-2"
      +"excerpt": "Laravel 10"
      +"description": "Laravel 10"
      +"is_published": 1
      +"min_to_read": 3
      +"created_at": null
      +"updated_at": null
    }
    1 => {#310 ▶
      +"id": 4
      +"title": "title 4"
      +"slug": "title-4"
      +"excerpt": "excerpt 4"
      +"description": "description 4"
      +"is_published": 1
      +"min_to_read": 2
      +"created_at": null
      +"updated_at": null
    }
    2 => {#308 ▶
      +"id": 6
      +"title": "title 6"
      +"slug": "title-6"
      +"excerpt": "excerpt 6"
      +"description": "description 6"
      +"is_published": 1
      +"min_to_read": 3
      +"created_at": null
      +"updated_at": null
    }
    3 => {#309 ▶
      +"id": 7
      +"title": "X"
      +"slug": "X"
      +"excerpt": "excerpt"
      +"description": "description"
      +"is_published": 1
      +"min_to_read": 2
      +"created_at": null
      +"updated_at": null
    }
  ]
  #escapeWhenCastingToString: false
}
```

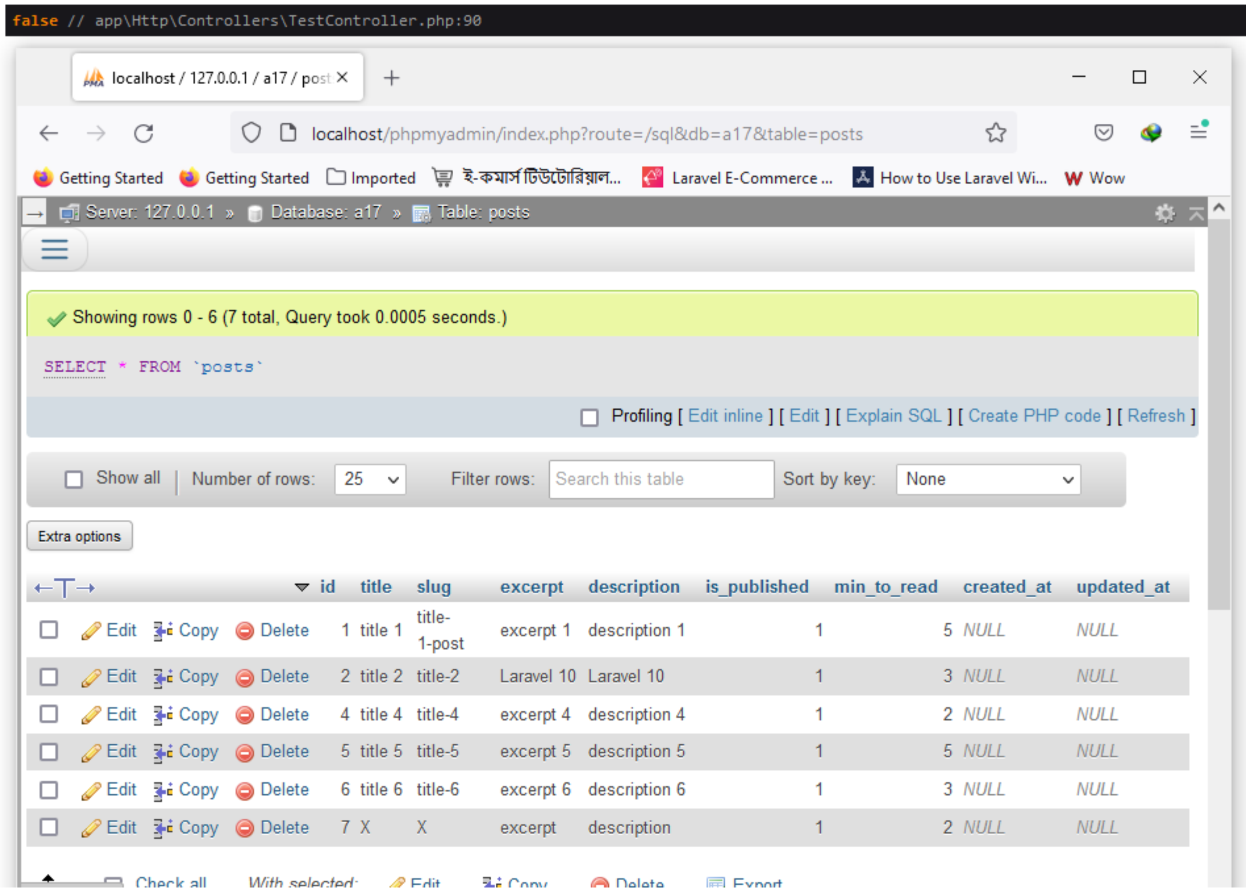
Fig: Output of whereNot() code snippet

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?

Answer: The exists() method returns true if at least one record matches the query criteria and false otherwise. This method can be used to check if any records exist that satisfy specific conditions.



```php
public function question13(){
    $hasPost = DB::table('posts')->where('id',3)->exists();
    dd($hasPost);
}
```



```
false // app\Http\Controllers\TestController.php:90
```

localhost / 127.0.0.1 / a17 / post
localhost/phpmyadmin/index.php?route=/sql&db=a17&table=posts
Getting Started
Getting Started
Imported
ই-কমার্স টিউটোরিয়াল...
Laravel E-Commerce ...
How to Use Laravel Wi...
Wow
Server: 127.0.0.1 » Database: a17 » Table: posts

Showing rows 0 - 6 (7 total, Query took 0.0005 seconds.)

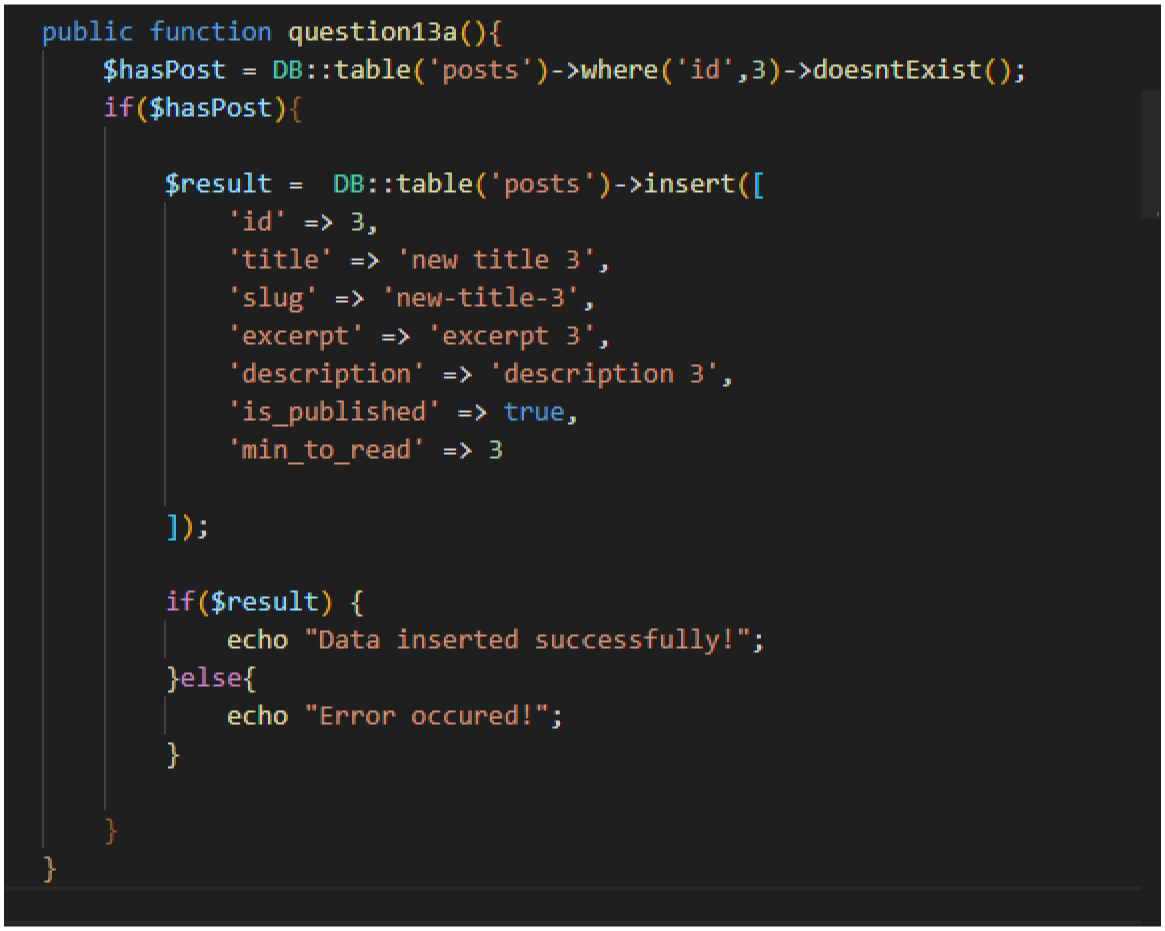
```sql
SELECT * FROM `posts`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	id	title	slug	excerpt	description	is_published	min_to_read	created_at	updated_at
Edit Copy Delete	1	title 1	title-1-post	excerpt 1	description 1	1	5	NULL	NULL
Edit Copy Delete	2	title 2	title-2	Laravel 10	Laravel 10	1	3	NULL	NULL
Edit Copy Delete	4	title 4	title-4	excerpt 4	description 4	1	2	NULL	NULL
Edit Copy Delete	5	title 5	title-5	excerpt 5	description 5	1	5	NULL	NULL
Edit Copy Delete	6	title 6	title-6	excerpt 6	description 6	1	3	NULL	NULL
Edit Copy Delete	7	X	X	excerpt	description	1	2	NULL	NULL

In this example we can see the query try to check if any posts record with id of 3 exists or not. In the output we can show as there is no such id so exists() return false.

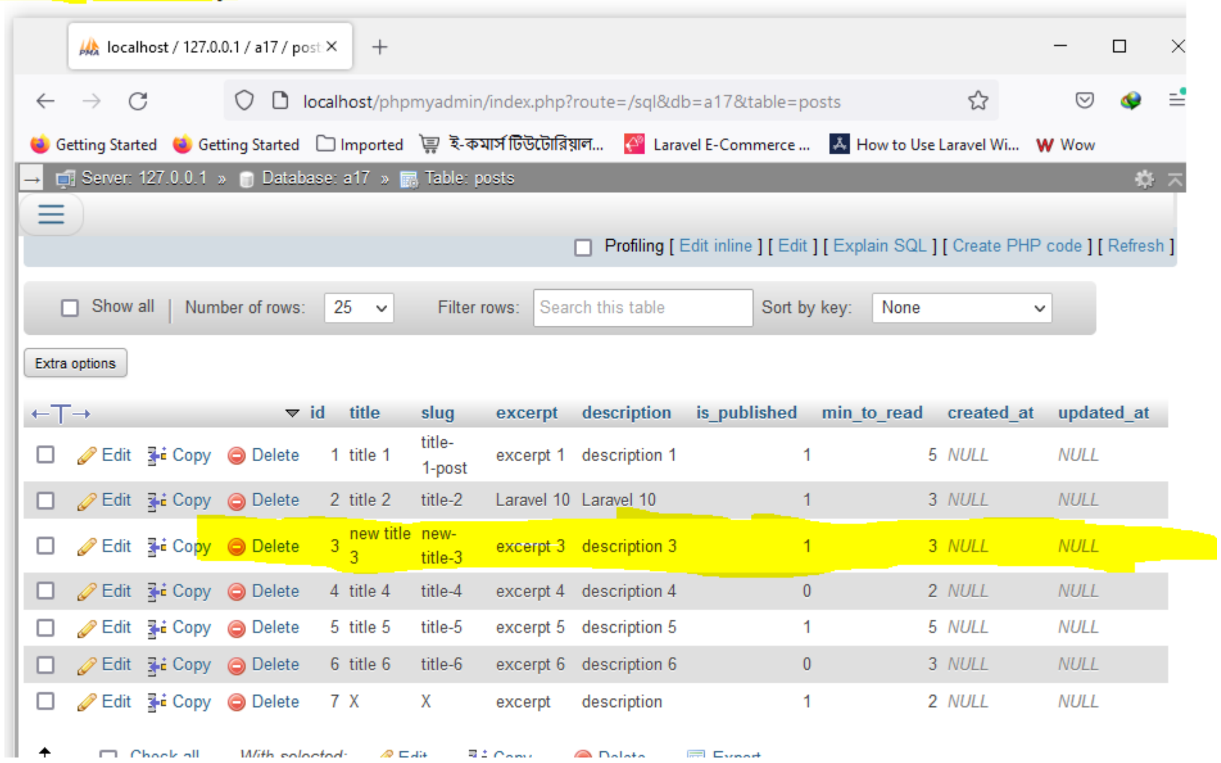
The **doesntExist()** method returns true if no records match the query criteria, and false if there is at least one matching record. It is the negation of the exists() method.



```php
public function question13a(){
    $hasPost = DB::table('posts')->where('id',3)->doesntExist();
    if($hasPost){

        $result =  DB::table('posts')->insert([
            'id' => 3,
            'title' => 'new title 3',
            'slug' => 'new-title-3',
            'excerpt' => 'excerpt 3',
            'description' => 'description 3',
            'is_published' => true,
            'min_to_read' => 3

        ]);

        if($result) {
            echo "Data inserted successfully!";
        }else{
            echo "Error occured!";
        }

    }
}
```

In this example we try to insert new records if id of 3 does not exists in posts table. As we don't have id of '3' in posts table as shown in previous example so **doesntExist()** return true hence new data is inserted with id of '3'.

Data inserted successfully!


localhost / 127.0.0.1 / a17 / post

localhost/phpmyadmin/index.php?route=/sql&db=a17&table=posts

Server: 127.0.0.1 » Database: a17 » Table: posts

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

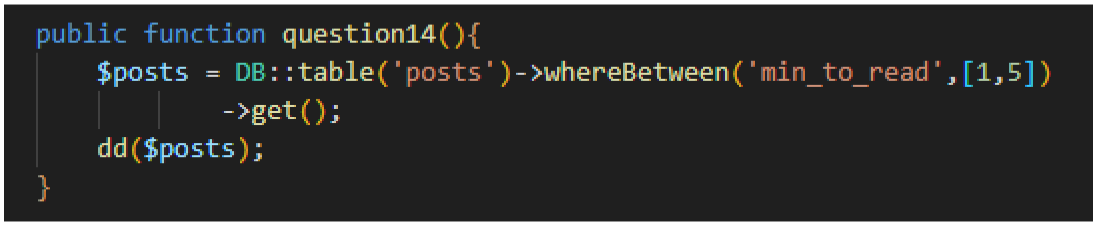
Extra options

				id	title	slug	excerpt	description	is_published	min_to_read	created_at	updated_at
☐	Edit	Copy	Delete	1	title 1	title-1-post	excerpt 1	description 1	1	5	NULL	NULL
☐	Edit	Copy	Delete	2	title 2	title-2	Laravel 10	Laravel 10	1	3	NULL	NULL
☐	Edit	Copy	Delete	3	new title 3	new-title-3	excerpt 3	description 3	1	3	NULL	NULL
☐	Edit	Copy	Delete	4	title 4	title-4	excerpt 4	description 4	0	2	NULL	NULL
☐	Edit	Copy	Delete	5	title 5	title-5	excerpt 5	description 5	1	5	NULL	NULL
☐	Edit	Copy	Delete	6	title 6	title-6	excerpt 6	description 6	0	3	NULL	NULL
☐	Edit	Copy	Delete	7	X	X	excerpt	description	1	2	NULL	NULL

Figure: output of **doesntExist**() code snippet

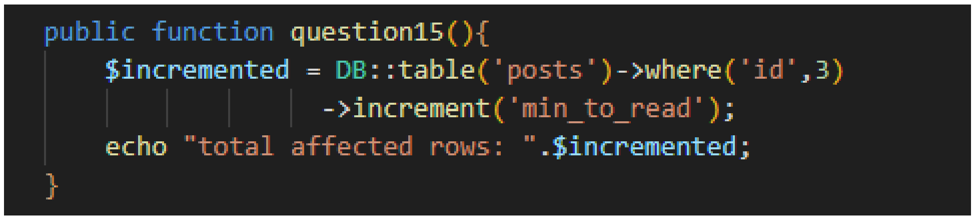
14.Write the code to retrieve records from the "posts" table where the "min_to_read" column is between 1 and 5 using Laravel's query builder. Store the result in the $posts variable. Print the $posts variable.

Answer:


```
public function question14(){
    $posts = DB::table('posts')->whereBetween('min_to_read',[1,5])
            ->get();
    dd($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.

Answer:



```php
public function question15(){
    $incremented = DB::table('posts')->where('id',3)
                ->increment('min_to_read');
    echo "total affected rows: ".$incremented;
}
```