OSTAD.APP

Web Development with PHP & Laravel - Batch 1

Name of Assignment: Query Builder in Laravel

Full Name: Md. Rafiqul Islam

Module: 17

(01) Laravel's query builder is a fluent interface for creating and executing database queries in Laravel. It provides a simple and elegant way to interact with databases by allowing developers to build queries using chainable methods. The query builder abstracts the underlying database engine and provides a consistent API for performing CRUD (Create, Read, Update, Delete) operations. It supports various features such as selecting columns, joining tables, applying conditions, sorting, and grouping results. The query builder also helps in preventing SQL injection by automatically binding parameters.

(02) Please check the following codes:

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

dd($posts);
```

(03) The distinct() method in Laravel's query builder is used to retrieve only distinct (unique) values from a column in a query result. It ensures that the result set does not contain duplicate values for the specified column. The distinct() method is typically used in conjunction with the select() method to specify the columns to select and retrieve unique values from.

Example:

```
$uniqueTitles = DB::table( table: 'posts')
    ->select( columns: 'title')
    ->distinct()
    ->get();

dd($uniqueTitles);
```

(04) Please check the following codes:

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

dd($posts->description);
```

(05) Please check the following codes:

```
$posts = DB::table( table: 'posts')
    ->where( column: 'id', operator: 2)
    ->value( column: 'description');

dd($posts);
```

(06) The first() method is used to retrieve the first record that matches the query conditions, while the find() method is used to retrieve a record by its primary key.

The first() method retrieves the first record based on the query conditions and returns an instance of the query builder, allowing you

to access the columns of the record using property syntax or methods.

The find() method retrieves a record by its primary key. It expects the primary key value as an argument and returns the first matching record. The record is returned as an instance of the query builder, and you can access its columns using property syntax or methods.

Example of first():

```php
$post = DB::table( table: 'posts')
    ->where( column: 'is_published', operator: true)
    ->orderBy( column: 'created_at')
    ->first();

dd($post);
```

Example of find():

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

dd($post);
```

(07) Please check the following codes:

```php
$posts = DB::table( table: 'posts')
    ->pluck( column: 'title');

dd($posts);
```

(08) Please check the following codes:

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

dd($result);
```

(09) Please check the following codes:

```php
$affectedRows = DB::table( table: 'posts')
    ->where( column: 'id', operator: 2)
    ->update([
        'excerpt' => 'Laravel 10',
        'description' => 'Laravel 10'
    ]);

dd($affectedRows);
```

(10) Please check the following codes:

```php
$affectedRows = DB::table( table: 'posts')
    ->where( column: 'id', operator: 3)
    ->delete();

dd($affectedRows);
```

(11) The aggregate methods in Laravel's query builder are used to perform calculations on a set of values. They allow you to retrieve aggregate information about a column, such as counting the number of rows, calculating the sum, average, maximum, or minimum value.

Examples:

- count():

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

dd($totalPosts);
```

- sum():

```php
$totalViews = DB::table( table: 'posts')->sum( column: 'views');

dd($totalViews);
```

- avg():

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

dd($averageRating);
```

- max():

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

dd($highestPrice);
```

- min():

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

dd($lowestStock);
```

(12) The whereNot() method in Laravel's query builder is used to add a "where not" condition to a query. It allows you to specify a column and a value that the column should not match.

Example:

```php
$users = DB::table( table: 'users')
    ->whereNot( column: 'status', operator: 'active')
    ->get();

dd($users);
```

(13) The exists() and doesntExist() methods in Laravel's query builder are used to check the existence of records in a table.

The exists() method returns true if there are any records that match the query conditions and false otherwise.

The doesntExist() method returns true if there are no records that match the query conditions and false if there is at least one matching record.

Example of exists():

```php
$exists = DB::table( table: 'users')
    ->where( column: 'email', operator: 'rafiqulislam474@gmail.com')
    ->exists();

dd($exists);
```

Example of doesntExist():

```
$doesntExist = DB::table( table: 'users')
    ->where( column: 'email',  operator: 'john@example.com')
    ->doesntExist();

dd($doesntExist);
```

(14) Please check the following codes:

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

dd($posts);
```

(15) Please check the following codes:

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

dd($affectedRows);
```