
Questions and Answers:

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

database query builder in Laravel provides an easy interface to create and run database queries. It can be used to perform most database operations, from basic DB Connection, CRUD, Aggregates, etc. and it works on all supported database systems.

The notable factor about query builder is that, since it uses the PHP Data Objects (PDO), It inherently provides protection for applications against SQL injection attacks. There is no need to clean or sanitize strings passed to the query builder as query bindings.

Example:

```
$users = DB::table('users')->get();
```

`DB::table` is responsible to begin a fluent query against a database table. The table from which the value has to be selected is mentioned inside the brackets within quotes and finally the `get()` method gets the values. Query builder allows the developer to search and filter database objects, select objects and columns, create relationships between objects, view formatted query results, and save queries with little or no SQL knowledge.

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.

```
app\Http\Controllers\PostController:
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function getExcerptAndDescription()
    {
        $posts = DB::table('posts')->select('excerpt', 'description')->get();

        foreach ($posts as $post) {
            echo "Excerpt: " . $post->excerpt . "\n";
            echo "Description: " . $post->description . "\n";
        }
    }
}
```

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

In Laravel, the `distinct()` method is used to fetch distinct records from the database, it is also part of Laravel query builder, which means it can be chained to other query builder methods as well. The typical usage of this method is to find how many users made comments on a post.

For example:

```
$uniqueTitles = DB::table('posts')->select('title')->distinct()->get();
foreach ($uniqueTitles as $post) {
    echo $post->title . "\n";
}
```

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.

```
app\Http\Controllers\PostController:
```

```
<?php
namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function getFirstRecordWithId()
    {
        $posts = DB::table('posts')->where('id', 2)->first();

        if ($posts) {
            echo "Description: " . $posts->description;
        } else {
            echo "post not found.";
        }
    }
}
```

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.

app\Http\Controllers\PostController:

```
<?php
namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function getRecordWithId()
    {
        $posts = DB::table('posts')->where('id', 2)->get();

        foreach ($posts as $post) {
            echo $post->description;
        }
    }
}
```

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

The `first()` method is used to retrieve the first record that matches the specified conditions or the first record in the table if no conditions are specified. It returns a single object representing the retrieved record.

The `find()` method is used to retrieve a record by its primary key value. It expects the primary key value as an argument and returns a single object representing the retrieved record. If the record with the specified primary key value is not found, it returns null.

Using `first()` and `find()` together helps to first retrieve a single record based on certain conditions, and then retrieve the complete record using its primary key value.

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.

app\Http\Controllers\PostController:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function getTitleColumn()
    {
        $posts = DB::table('posts')->pluck('title');

        foreach ($posts as $title) {
            echo $title . "\n";
        }
    }
}
```

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.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function insertNew()
    {
        $data = [
            'title' => 'X',
            'slug' => 'X',
            'excerpt' => 'excerpt',
            'description' => 'description',
            'is_published' => true,
            'min_to_read' => 2,
        ];

        $result = DB::table('posts')->insert($data);

        echo "Result: " . ($result ? "OK" : "Error");
    }
}
```

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.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function updateExcerptAndDescriptionWithId()
    {
        $data = [
            'excerpt' => 'Laravel 10',
            'description' => 'Laravel 10',
        ];

        $rowsChanged = DB::table('posts')->where('id', 2)->update($data);

        echo "Number of affected rows: " . $rowsChanged;
    }
}
```

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.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function deleteWithId()
    {
        $result = DB::table('posts')->where('id', 3)->delete();

        echo "Number of affected rows: " . $result;
    }
}
```

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.

The query builder provides a variety of methods for retrieving aggregate values.

Examples:

```
use Illuminate\Support\Facades\DB;

$numberOfOrders = DB::table('orders')->count();

$totalPrice = DB::table('orders')->sum('price');

$averagePriceOfOrders = DB::table('orders')->avg('price');

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

$minPrice = DB::table('orders')->min('price');
```

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

The `whereNot()` method may be used to negate a given group of query constraints. For example, the following query excludes products that are on clearance or which have a price that is less than 10:

```
$products = DB::table('products')
    ->whereNot(function (Builder $query) {
        $query->where('clearance', true)
        ->orWhere('price', '<', 10);
    })
    ->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?

Laravel has two more eloquent methods `exists()` and `doesntExist()` for check if record exists in database table or not. `exists()` and `doesntExist()` methods return true or false value so they can be just put in the condition.

`exists()` Example:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\User;

class UserController extends Controller
{

    public function index()
    {
        $isExist = User::select("*")
            ->where("email", "alice@names.org")
            ->exists();

        if ($isExist) {
            dd('Record is available.');
```

```
        }else{
            dd('Record is not available.');
```

```
        }
```

```
    }
```

`doesntExist()` Example:

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
```

```
use App\User;
```

```
class UserController extends Controller
```

```
{
```

```
    public function index()
```

```
    {
```

```
        $isExist = User::select("*")
```

```
            ->where("email", "bob@names.org")
```

```
            ->doesntExist();
```

```
        if ($isExist) {
```

```
            dd('Record is not available.');
```

```
        }else{
```

```
            dd('Record is available.');
```

```
        }
```

```
    }
```

```
}
```

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.

```
app\Http\Controllers\PostController:
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function getWithMinToRead()
    {
        $posts = DB::table('posts')->whereBetween('min_to_read', [1, 5])->get();

        foreach ($posts as $post) {
            echo "ID: " . $post->id . "\n";
            echo "Title: " . $post->title . "\n";
            echo "Min To Read: " . $post->min_to_read . "\n";
        }
    }
}
```

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.

app\Http\Controllers\PostController:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class PostController extends Controller
{
    public function inceaseMinToRead()
    {
        $affectedRows = DB::table('posts')->where('id', 3)->increment('min_to_read');

        echo "Number of affected rows: " . $affectedRows;
    }
}
```