Assignment of Module-17

Question: 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 one of the most popular features of the Laravel framework that delivers a suitable and expressive way to interact with databases, without having to worry about the underlying SQL.
One of the main advantages of using Laravel's query builder is its simplicity and elegance. It offers a more legible and intuitive syntax compared to writing raw SQL queries.

How laravel query provides a simple and elegant way to interact with databases:
To interact with databases Laravel provides a simple and elegant way through its query builder and Eloquent ORM.
The query builder provides a fluent interface for building database queries. This means that you can chain together methods to build complex queries without having to worry about the underlying SQL.
For example, the following code will select all users from the users table:
$users = DB::table('users')->get();

Another example code to get a single user from database by their ID:
$user = DB::table('users')->find(1);

Here are some ways Laravel's query builder and Eloquent ORM simplify database interactions:
Fluent Query Builder,
Object-Oriented Model Relationships,
Database Migrations,
Database Abstraction,
Query Caching,
Eloquent ORM 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.

Answer :

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

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

return $posts;
```

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

Answer:

Using distinct() method, we retirve the unique values from the particular column or set of columns in database table.
This means that if there are multiple rows in the table with the same value for a specific column, only one row will be returned.
How distinct() is used in conjunction with the select() method:

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

$uniqueValues = DB::table('users')
   ->select('name')
   ->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.

Answer :

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

echo $posts->description;
```

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.

Answer :

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

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

return $posts->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?

Answer:

The first() and find() methods in Laravel's query builder are both used to retrieve single records from a database. However, there are some fundamental differences between the two methods.

first() :

The first() method returns the first record that matches the given criteria. If there are no records that match the criteria, then first() will return null.

example code:

```
$post = DB::table('posts')->first();
```

find():

The find() method returns all records that match the given criteria. If there are no records that match the criteria, then find() will return an empty collection.

example code:

```
$post = DB::table('posts')->find(1);
```

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.

Answer:

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

$posts = DB::table('posts')
    ->select('title')
    ->get();

return $posts;
```

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.

Answer:

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

$result = DB::table('posts')->insert([
    'title' => 'X',
    'slug' => 'X',
    'excerpt' => 'excerpt',
    'description' => 'description',
    'is_published' => true,
    'min_to_read' => 2,
]);
if ($result) {
    echo "The record has been inserted successfully.";
} else {
    echo "The record could not be inserted.";
}
```

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.

Answer:

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

$query = DB::table('posts')
   ->where('id', 2)
   ->update([
      'excerpt' => 'Laravel 10',
      'description' => 'Laravel 10 as description.',
   ]);

if($query){
   return "The records have been updated";
}
```

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.

Answer:

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

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

if($query){
   return "The record has been deleted";
}
```

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.

Answer:

count():
count() returns the number of rows in the result set. It returns the count as an integer value.
Example:
$count = DB::table('products')->count();

sum():
sum() returns the sum of all the values in a column. It returns the sum as a numeric value.
Example:
$total_price = DB::table('products')->sum('price');

avg():
The avg() returns the average of all the values in a column in the database table. It returns the average as a numeric value.
Example:
$average_price = DB::table('products')->avg('price');

max():
The max() method retrieves the maximum value from a specific column in the database table. It returns the maximum value.
Example:
$average_price = DB::table('products')->max('price');

min():
The min() method retrieves the minimum value from a specific column in the database table. It returns the minimum value.
Example:
$average_price = DB::table('products')->min('price');

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

Answer:

The whereNot() method is used to filter the results of a Laravel query builder by excluding rows that match a certain condition. That means the whereNot() method is used to add a "not equal" condition to a query. It allows us to retrieve records that do not match a specific value or condition.

Example:

```
$result =  DB::table('products')
      ->whereNot('products.price', '=', 2000)
      ->get();

 return $result;
```

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?

Answer:

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

-The exists() method returns true if any records exist that match the query's constraints, and false otherwise.

-The doesntExist() method returns true if no records exist that match the query's constraints, and false otherwise.

Example of exists():

```
$exists = DB::table('users')
  ->where('email', 'user@example.com')
  ->where('role', 'admin')
  ->exists();
```

Example of doesntExist():

```
$exists = DB::table('users')
  ->where('email', 'user@example.com')
  ->where('role', 'admin')
  -doesntExist();
```

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.

Answer:

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

return $posts;
```

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.

Answer:

```
$incrementedRows = DB::table('posts')
   ->where('id', 3)
   ->increment('min_to_read', 1);

return $incrementedRows
```