Module-17 Query Builder in Laravel Assignment-17:

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

Laravel's query builder is a fluent interface for building and executing database queries in Laravel. It provides a simple and elegant way to interact with databases by allowing developers to construct queries using chainable methods instead of writing raw SQL. The query builder abstracts the underlying database system and provides a consistent API for performing common database operations, such as selecting, inserting, updating, and deleting records.

Task 2: EWrite 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.

Code to retrieve the "excerpt" and "description" columns from the "posts" table using Laravel's query builder:

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

Task 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 Laravel's query builder is used to retrieve only unique values from a column in the database table. It ensures that the query result does not contain any duplicate values for the specified column.When used in conjunction with the select() method, the distinct() method allows you to specify the column(s) for which you want to retrieve unique values. It modifies the query to include the DISTINCT keyword in the generated SQL query.

Here's an example to illustrate the usage of distinct() in conjunction with select(): 1. `shell`
`$uniqueCategories = DB::table('products') ->select('category') ->distinct() ->get();`

Task 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.

Code to retrieve the first record from the "posts" table where the "id" is 2 using Laravel's query builder:

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

Task 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.

Code to retrieve the "description" column from the "posts" table where the "id" s 2 using Laravel's query builder:

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

Task 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 in Laravel's query builder retrieves the first record that matches the query criteria, whereas the find() method retrieves a single record by its primary key. first() is used when you want to retrieve the first matching record based on the query conditions, while find() is used when you know the primary key value and want to retrieve a specific record.

1. Using first()
2. `$firstUser = DB::table('users')->first();`
3. Using find()
4. `$userId = 1;`
5. `$user = DB::table('users')->find($userId);`
6. ````` ``` `````

Task 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.

Code to retrieve the "title" column from the "posts" table using Laravel's query builder:

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

Task 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.

Code to insert a new record into the "posts" table using Laravel's query builder:

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

Task 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.

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:

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

Task 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.

Code to delete the record with the "id" of 3 from the "posts" table using Laravel's query builder:

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

Task 11: Explain the purpose and usage of the aggregate methods ount(), sum(), avg(), max(), and min() in Laravel's query builder. Provide an example of each.

The aggregate methods (count(), sum(), avg(), max(), and min()) in Laravel's query builder are used to perform calculations on a set of database records. These methods allow you to retrieve aggregated data from the database vithout having to process the records manually. Examples:

1. count() method example:
2. `$totalPosts = DB::table('posts')->count();`
   `echo $totalPosts;`
3. sum() method example:
4. `$totalViews = DB::table('posts')->sum('views');`
   `echo $totalViews;`
5. avg() method example:
6. `$averageRating = DB::table('ratings')->avg('rating');`
   `echo $averageRating;`
7. max() method example:
8. `$highestSalary = DB::table('employees')->max('salary');`
   `echo $highestSalary;`
9. min() method example:
10. `$lowestAge = DB::table('users')->min('age');`
    `echo $lowestAge;`

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

The whereNot() method in Laravel's query builder is used to add a "WHERE NOT" condition to a query. It excludes the records that match the given ondition. Example usage:

```
$posts = DB::table('posts')
->whereNot('category', 'news')
->get();
print_r($posts);
```

his query will retrieve all posts where the category is not "news".

Task 13: Explain the difference between the exists() and loesntExist() methods in Laravel's query builder. How are they ısed to check the existence of records?

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

xists() returns true if any record matches the query conditions and false otherwise. Example usage:

```
$exists = DB::table('posts')
->where('category', 'news')
->exists();
echo $exists;
```

doesntExist() returns true if no record matches the query conditions and false if at least one record exists. Example usage:

```
$doesntExist = DB::table('posts')
->where('category', 'news')
->doesntExist();
echo $doesntExist;
```

Task 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.

Code to retrieve records from the "posts" table where the "min_to_read" column is between 1 and 5 using Laravel's query builder:

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

Task 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.

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:

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

Author: Marajul Islam