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

Ans:
Laravel's query builder is a feature provided by the Laravel framework that allows developers to build database queries using a fluent, intuitive syntax. It provides a convenient and expressive way to interact with databases without writing raw SQL statements.
The query builder allows us to perform common database operations such as selecting, inserting, updating, and deleting records. It supports various database systems, including MySQL, PostgreSQL, SQLite, and SQL Server. The query builder abstracts the underlying database system, making it easier to switch between different database engines without having to change our code.

Some key aspects of Laravel's query builder and how it simplifies database interactions:

a) Fluent Interface: The query builder uses a fluent interface, which means we can chain methods together to construct queries. This results in a more readable and concise code compared to writing SQL queries manually.

b) Selecting Data: We can specify the columns we want to retrieve from the table using the 'select' method. We can also use methods like 'where', 'orWhere', 'whereIn', 'orWhereIn', etc., to add conditions to our queries.

c) Joining Tables: Laravel's query builder allows we to perform various types of table joins, such as inner join, left join, and right join. The 'join' method enables we to specify the table to join and the columns to join on.

d) Inserting and Updating Data: The query builder provides methods like 'insert', 'update', and 'delete' for performing data manipulation operations. We can pass an array of data to be inserted or updated, making it straightforward to work with multiple records.

e) Pagination: Laravel's query builder integrates seamlessly with the framework's pagination features. We can use the 'paginate' method to retrieve paginated results from a query.

f) Raw Expressions: If we need to include raw SQL expressions in our queries, the query builder allows we to use the 'selectRaw', 'whereRaw', and similar methods to write custom SQL code within our queries.

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.

Ans:

```
16
17  Route::get('/', [AssignmentController::class, 'ExcerptDes']);
18
```

Image (Route): Task 2

```
1 reference | 0 overrides
11      function ExcerptDes(){
12          $posts = DB::table('posts')->select('excerpt', 'description')->get();
13          return $posts;
14      }
15
```

Image (Method): Task 2

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

Ans:
In Laravel's query builder **distinct()** method is used to retrieve unique records from a database table.

When used in conjunction with the **select()** method, the **distinct()** method filters the selected columns to return only unique values. It filters out any duplicate rows based on the selected columns.
For example:

```php
//Task 3:
1 reference | 0 overrides
function DistinctTitle(){
    $posts = DB::table('posts')->select('title')->distinct()->get();
    return $posts;
}
```

Image (Method): Task 3

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.

Ans:

```php
// Task: 4:
Route::get('/firstrecord', [AssignmentController::class, 'FirstRecord']);
```

Image (Route): Task 4

```php
// Task 4:
0 references | 0 overrides
function FirstRecord(){
    $posts = DB::table('posts')->where('id', '=', 2)->first();

    return $posts->description;
}
```

Image (Method): Task 4

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.

Ans:

```php
// Task: 5:
Route::get('/posts-des', [AssignmentController::class, 'DesFromPost']);
```

Image (Route): Task 5

```php
1 reference | 0 overrides
function DesFromPost(){
    $posts = DB::table('posts')->where('id', 2)->pluck('description');
    return $posts;
}
```

Image (Method): Task 5

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.

Ans:
In Laravel's query builder **distinct()** method is used to retrieve unique records from a database table.

When used in conjunction with the **select()** method, the **distinct()** method filters the selected columns to return only unique values. It filters out any duplicate rows based on the selected columns.
For example: