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

Image (Method): Task 5

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

Ans:
In Laravel's query builder, differences between the first() and find() methods are used to retrieve single records from a table, but they differ in how they are used.

first() method: In this method is used to retrieve the first record that matches the query conditions. It returns a single object containing the record's data.
Example:

```php
function DiffFirst(){
    $posts = DB::table('posts')->first();
    return $posts;
}
```

Image: Task 6 first() method

find() method: This method used to retrieve a record based on its primary key value. It assumes that the primary key column in the table is named "id" by default.
Example:

```php
function DiffFind(){
    $posts = DB::table('posts')->find(3);
    return $posts;
}
```

Image: Task 6 find() method

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.

Ans:

```php
// Task: 7:
Route::get('/posts-title', [AssignmentController::class, 'PostTitle']);
```

Image (Route): Task 7

```php
// Task 7:
1 reference | 0 overrides
function PostTitle(){
    $posts = DB::table('posts')->select('title')->get();
    return $posts;
}
```

Image (Method): Task 7

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.

Ans:

```php
// Task: 8:
Route::get('/insert-new', [AssignmentController::class, 'InsertNew']);
```

Image (Route): Task 8

Continued for Q 8:

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

Image (Method): Task 8

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.

Ans:

```php
// Task: 9:
Route::get('/updatedata', [AssignmentController::class, 'UpdateData']);
```

Image (Route): Task 9

```php
function UpdateData() {
    $DataUpdate = DB::table( 'posts' )
    ->where('id', 2)
    ->update([
        'excerpt' => 'Laravel 10',
        'description' => 'Laravel 10'
    ]);
    return $DataUpdate;
}
```

Image (Method): Task 9

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.

Ans:

```php
// Task: 10:
Route::get('/deletedata', [AssignmentController::class, 'DeleteData']);
```

Image (Route): Task 10

```php
function DeleteData() {
    $delete = DB::table( 'posts' )
    ->where('id', 3)
    ->delete();
    return "Affected Rows are: " .$delete;
}
```

Image (Method): Task 10

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.

Ans:
In Laravel's query builder, the purpose of the aggregate each methods explained below with examples.

count() method: This method is used to retrieve the total count of records that match a specific condition. It returns the number of rows in the result set.
Example:

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

Image (Method): count()

sum() method: This method is used to calculates the sum of a specific column's values. It is typically used on numeric columns.
Example:

```php
$sum = DB::table( 'orders' )->sum( 'price' );
return $sum;
```

Image (Method): sum()

avg() method: This method is used to calculates the average value of a specific column. It is commonly used on numeric columns.
Example:

```php
$avg = DB::table( 'orders' )->avg( 'ratings' );
return $avg;
```

Image (Method): avg()

max() method: This method is used to retrieves the maximum value from a specific column. It is typically used on either numeric or date or time columns.
Example:

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

Image (Method): max()

min() method: This method is used to retrieves the minimum value from a specific column. It is typically used on either numeric or date or time columns.
Example:

```php
$min = DB::table( 'products' )->min( 'price' );
return $min;
```

Image (Method): min()

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

Ans:
In Laravel's query builder, the **whereNot()** method is used to add a "not equal" condition to the query. It allows you to retrieve records where a specific column's value is not equal to a given value.
Example:

```php
$affected = DB::table( 'products' )->whereNot( 'title', 'LIKE', '%Car%' )
->get();
return $affected;
```

Image (Method): Task 12

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?

Ans:
In Laravel's query builder, the exists() and doesntExist() methods are used to check the existence of records in a table.

exists() method: The **exists()** method is used to check if any records exist in a table that match a specific condition. It returns **true** if at least one record is found, and **false** otherwise.
Example:

```php
$postPublished = DB::table( 'posts' )->where( 'is_published', true )->exists();
```

Image : exists() method, Task 13

doesntExists() method: The doesntExist() method is the opposite of exists() method. It is used to check, if no records exist in a table that match a specific condition. It returns **true** if no records are found, and **false** otherwise.
Example:

```php
$postNotPublished = DB::table( 'posts' )->where( 'is_published', false )
->doesntExist();
```

Image: doesntExist() method, Task 13

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.

Ans:

```php
// Task: 14:
Route::get('/minsread', [AssignmentController::class, 'MinutesRead']);
```

Image(Route): Task 14

```php
1 reference | 0 overrides
function MinutesRead() {
    $minReadBetween = DB::table( 'posts' )
    ->whereBetween( 'min_to_read', [1, 5] )
    ->get();
    return $minReadBetween;
}
```

Image(Method): Task 14

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.

Ans:

```php
// Task: 15:
Route::get('/incrementreadtime', [AssignmentController::class, 'IncReadTime']);
```

Image(Route): Task 15

```php
1 reference | 0 overrides
function IncReadTime() {
    $incReadTime = DB::table( 'posts' )->where( 'id', 3 )
    ->increment('min_to_read', 1);

    return $incReadTime;
}
```

Image(Method): Task 15