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 a feature of the Laravel framework that provides a simple and elegant way with wide range of methods to interact with databases. It allows developers to build database queries using a fluent, chainable interface instead of writing raw SQL queries which is flexible to use.

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: To retrieve the "excerpt" and "description" columns from the "posts" table using Laravel's query builder and store the result in the $posts variable, then print the $posts variable, we have to write the following code:

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

print_r($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: The distinct() method in Laravel's query builder is used to retrieve only unique values from a column. It ensures that the result set contains no duplicate values. It is used in conjunction with the select() method to specify the columns to be selected, and it modifies the query to return only distinct values for those columns. This is useful when you want to eliminate duplicate records from the result set.

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: 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, and print the "description" column of the $posts variable:

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

if ($posts) {

   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: 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, and print the $posts variable:

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

print_r($posts);
```

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: In Laravel's query builder, the first() method is used to retrieve the first record that matches the query conditions, while the find() method is used to retrieve a record by its primary key value.

The first() method returns a single object representing the first record, or null if no record is found. The find() method returns a single object representing the record with the specified primary key, or null if the record is not found.

The first() method is typically used when you need to retrieve the first matching record based on certain conditions, while the find() method is used when you know the primary key value of the record you want to retrieve.

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: Code to retrieve the "title" column from the "posts" table using Laravel's query builder, store the result in the $posts variable, and print the $posts variable:

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

print_r($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: 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, then print the result of the insert operation:

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

print_r($result);
```

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: 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', and print the number of affected rows:

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

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: Code to delete the record with the "id" of 3 from the "posts" table using Laravel's query builder, and print the number of affected rows:

```
$affectedRows = DB::table('posts')->where('id', 3)->delete();
echo "Number of affected rows: " . $affectedRows;
```

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: The aggregate methods in Laravel's query builder are used to perform calculations on a set of records and return a single result. Here are examples of each aggregate method:

- **count():** Returns the number of records in a table or the number of records that match certain conditions.
  Ex. *$totalCount = DB::table('posts')->count();*
- **sum():** Returns the sum of a column's values for a set of records.
  Ex. *$totalAmount = DB::table('orders')->sum('amount');*

- **avg():** Returns the average value of a column for a set of records.
  Ex. *$averageRating = DB::table('reviews')->avg('rating');*
- **max():** Returns the maximum value of a column for a set of records.
  Ex. *$maxMarks = DB::table('students')->max('marks');*
- **min():** Returns the minimum value of a column for a set of records.
  Ex. *$minAge = DB::table('users')->min('age');*

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

Answer: The whereNot() method in Laravel's query builder is used to add a "WHERE NOT" condition to the query. It excludes records that match the specified condition from the result set. Here's an example of its usage:

```
$posts = DB::table('posts')
    ->whereNot('status', 'published')
    ->get();
```

This example retrieves all posts where the "status" column is not equal to 'published'. Only the posts with a status other than 'published' will be included in the result set.

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 if any record exists or not.

exists(): Returns true if there is at least one record that matches the query conditions, and false otherwise. It is typically used in combination with other query builder methods to perform conditional operations.

Ex. *$exists = DB::table('posts')->where('status', 'published')->exists();*

doesntExist(): Returns true if no records match the query conditions, and false if there is at least one matching record. It is the opposite of the exists() method.

Ex. *$doesntExist = DB::table('posts')->where('status', 'published')->doesntExist();*

These methods are useful for checking if a record exists before performing an operation, such as inserting a new record or updating an existing one.

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: 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, and print the $posts variable:

```
$posts = DB::table('posts')
    ->whereBetween('min_to_read', [1, 5])
    ->get();
print_r($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: 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, and print the number of affected rows:

```
$affectedRows = DB::table('posts')
    ->where('id', 3)
    ->increment('min_to_read');
echo "Number of affected rows: " . $affectedRows;
```