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 powerful feature that allows developers to build and execute database queries using a fluent and expressive syntax. It provides a simple and elegant way to interact with databases by abstracting the underlying database engine and providing a consistent API for querying and manipulating data. With the query builder, developers can write database queries in a more readable and maintainable way compared to writing raw SQL queries. It allows you to perform common database operations such as selecting, inserting, updating, and deleting data with ease.

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
      $posts = DB::table('post')->pluck('experts','description');
      return $posts;
   }
}
```

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 rows from a query result set. It ensures that duplicate rows are eliminated, and only distinct values are returned. By default, when we use the select() method in Laravel's query builder, it retrieves all columns specified in the method call. However, in some cases, we may want to retrieve only unique values from a specific column or a combination of columns. That's why we use the distinct() method.

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
     $posts = DB::table('posts')->where('id','=','2')->first('description');
     return $posts ;
   }
}
```

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
     $posts = DB::table('posts')->where('id','=','2')->pluck('description');
     return $posts ;
   }
}
```

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

In Laravel's query builder, both the first() and find() methods are used to retrieve single records from a table, but they different in their behavior:

The first() method is used to retrieve the first record that matches the specified conditions. It is typically used when you want to fetch a single record based on certain criteria or without specifying a specific identifier.

The find() method is used to retrieve a record based on its primary key value. It expects a single argument, which is the primary key value of the record you want to retrieve. Suppose find(2) will return id no-2’s row.

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->pluck('title');
       return $posts;
   }
}
```

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->insert(
           [
               'title'=>'X',
               'slug'=>'X',
               'excerpt'=>'excerpt',
               'description'=>'description',
               'is_published'=>true,
               'min_to_read'=>2
           ]
       );
       print_r($posts) ;
   }
}
```

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->where('id','=','2')->update(
           [
               'excerpt'=>'Laravel 10',
               'description'=>'Laravel 10',
           ]
       );
       print_r($posts) ;
   }
}
```

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->where('id','=','3')->delete();
       print_r($posts) ;
   }
}
```

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.

The count() method is used to calculate the number of records that match a given condition or criteria.

The sum() method is used to calculate the sum of a specific column's values in a table.

The avg() method is used to calculate the average value of a specific column in a table.

The max() method is used to retrieve the maximum value of a specific column in a table.

The min() method is used to retrieve the minimum value of a specific column in a table.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $count= DB::table('products')->count();
       $sum=DB::table('products')->sum('price');
       $avg=DB::table('products')->avg('price');
       $max=DB::table('products')->max('price');
       $min=DB::table('products')->min('price');
       return
['count'=>$count,'sum'=>$sum,'avg'=>$avg,'max'=>$max,'min'=>$min];
   }
}
```

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 "not equal" condition to a query. It allows you to retrieve records that do not match a specific value or set of values.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->whereNot('id','>','2')->get();
       print_r($posts) ;

   }
}
```

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?

The exists() method is used to check if any records exist in the result set of a query. It returns true if there is at least one record that matches the specified conditions, and false otherwise.

The doesntExist() method is used to check if no records exist in the result set of a query. It returns true if no records match the specified conditions, and false if there is at least one matching record.

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->whereBetween('min_to_read',[1,5])->get();
       print_r($posts) ;

   }
}
```

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.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class demoController extends Controller
{
   function demoFunction(){
       $posts = DB::table('posts')->where('id','=','3')-
>increment('min_to_read');
       print_r($posts) ;
```

```
    }
}
```