Task 1:

Explanation:

Laravel's query builder is a database abstraction layer that allows developers to interact with databases using a simple and elegant API. It provides a fluent, chainable interface for building and executing database queries, eliminating the need for writing raw SQL statements.

With the query builder, you can perform various operations like selecting, inserting, updating, and deleting data. The fluent interface allows you to chain methods together, making the code concise and readable.

```
Route get( '/getExcerptDescription', [AssignmentController        class,

        'getExcerptDescription'] );
```

The query builder also ensures database agnosticism, meaning you can switch between different database systems without changing your code. It handles parameter binding, protecting against SQL injection attacks, and provides query logging and debugging capabilities.

Overall, Laravel's query builder simplifies database interactions by abstracting the complexity of SQL, offering a clean and intuitive syntax. It enhances productivity, improves code maintainability, and provides a convenient way to work with databases in Laravel applications.

Task 2:

Controller method:

```
function getExcerptDescription() {
$posts = DB table( 'posts' )
return $posts;
}
```

Route:

Task 3:

Describe:

The `distinct()` method in Laravel's query builder is used to retrieve only unique rows from a table. It ensures that the query results do not contain any duplicate rows.

When used in conjunction with the `select()` method, `distinct()` modifies the select statement to return only distinct (unique) values for the specified columns. It filters out any duplicate rows based on the selected columns.

Example:

```
$uniquePosts = DB table('posts') select('title') distinct() get();
```

Task 4:

Controller method:

```
function firstRecordDescription() {
$posts = DB table( 'posts' )
return $posts description;
}
```

Route:

```
Route get( '/firstRecordDescription', [AssignmentController class,
'firstRecordDescription'] );
```

Task 5:

Controller method:

```
function getDescription() {

$posts = DB table( 'posts' )

return $posts;

}
```

Route:

```
Route   get( '/getDescription', [AssignmentController     class, 'getDescription'] );
```

Task 6:

Explanation:

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

=> The `first()` method is used to retrieve the first record that matches the query conditions. It returns a single object containing the record's data.

Example:

$firstPost = DB table('posts') first();

=> The `find()` method is 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:

$post = DB table('posts') find(2);

Task 7:

.

Controller method:

function getAllTitle() {
$posts = DB table('posts')
return $posts;
}

Route:

Route get('/getAllTitle', [AssignmentController class, 'getAllTitle']);

Task 8:

Route:

Route post('/insertPost', [AssignmentController class, 'insertPost']);

Controller method:

```
function insertPost() {
$result = DB table( 'posts' ) insert( [
'title'     'X',
'slug'     'x',
'excerpt'          'excerpt',
'description'      'description',
'is_published'     true,
'min_to_read'    2,
] );
return $result;
}
```

Task 9:

Route:

```
Route    patch( '/update', [AssignmentController class, 'update'] );
```

Controller method:

```
function update() {

$affectedRows = DB table( 'posts' ) where( 'id', 2 )

update( [

'excerpt'          'Laravel 10',

'description'      'Laravel 10',

] );



return $affectedRows;

}
```

Task 10:

Route:

```
Route    delete( '/delete', [AssignmentController class, 'delete'] );
```

Controller method:

```
function delete() {
$affectedRows = DB table( 'posts' )
return $affectedRows;
}
```

Task 11:

Explanation:

In Laravel's query builder, the aggregate methods (count(), sum(), avg(), max(), and min()) are used to perform calculations on a specific column or set of columns in a table. These methods allow you to retrieve aggregated values and perform computations on your data. Here's an explanation and an example of each aggregate method:

01: count(): The count() 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:

```
$count = DB table( 'posts' ) count();
```

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

Example:

```
$sum = DB        table( 'orders' ) sum( 'amount' );
```

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

Example:

```
$avg = DB        table( 'orders' ) avg( 'amount' );
```

04: max(): The max() method retrieves the maximum value from a specific column. It is typically used on numeric or date/time columns.

Example:

```
$max = DB        table( 'orders' ) max( 'amount' );
```

05: min(): The min() method retrieves the minimum value from a specific column. It is commonly used on numeric or date/time columns.

Example:

```
$min = DB        table( 'orders' ) min( 'amount' );
```

Task 12:

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.
The whereNot() method is typically used in combination with the where() method to add additional conditions to the query. It accepts two parameters: the column name and the value to compare against.
Usage:

```
$posts = DB        table('posts')      whereNot('is_published', ' ', false)          get();
```

Task 13:

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

```
$hasPublishedPosts = DB table( 'posts' ) where( 'is_published', true ) exists();
```

02- doesntExist(): The doesntExist() method is the opposite of exists(). 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:

```
$noPublishedPosts = DBtable( 'posts' )    where( 'is_published', false )      doesntExist();
```

Task 14:

Route:

```
Route    get( '/getPostByMinRead', [AssignmentController          class, 'getPostByMinRead'] );
```

Controller method:

```
function getPostByMinRead() {
$posts = DB table( 'posts' )         whereBetween( 'min_to_read', [1, 5] )    get();
return $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.
Route:

```
Route    post( '/increments', [AssignmentController         class, 'increments'] );
```

Controller method:

```
function increments() {
$posts = DB table( 'posts' )         where( 'id', 3 )    increment( 'min_to_read', 1
```

```
);
return $posts;
}
```