

# Query Builder Methods

# DB facade

- The **DB** facade provides methods for each type of query: *select, update, insert, delete, and statement*.
- The facade utilizes the query builder to execute your database operations.
- <https://laravel.com/docs/11.x/queries>

# DB facade

The DB facade exposes many query builder methods:

- Select:** `DB::table('users')->get();`
- Insert:** `DB::table('users')->insert(['name' => 'John']);`
- Update:** `DB::table('users')->where('id', 1)->update(['active' => 0]);`
- Delete:** `DB::table('users')->delete();`
- Raw Queries:** `DB::statement('ALTER TABLE ...');`

# Query builder

- Does not automatically handle model relationships.

# Outline

1. Retrieving All Rows
2. Retrieving a specific row
3. Retrieve the values of a single column
4. Retrieve the values of multiple columns
5. Retrieving aggregate values
6. Pagination in Laravel
7. Raw Expressions
8. Retrieving a result of joined tables

1

```
SELECT * FROM table_name;
```

# Retrieving All Rows From A Table

- Go to `app/Http/Controllers/DiaryEntryController.php`

## Query Builder

```
use Illuminate\Support\Facades\DB;
```

```
public function display_diary()
{
    $userId = Auth::id(); // Get the authenticated user's ID

    // Fetch all diary entries for the authenticated user
    $diaryEntries = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->get();

    return view('diary.display_diary', compact('diaryEntries'));
}
```



Use the `table` method on the DB facade to specify the table name and then use the `get` method to retrieve all records

## Eloquent

```
public function index()
{
    $diaryEntries = Auth::user()->diaryEntries()->get();
    return view('diary.index', compact('diaryEntries'));
}
```

2

```
SELECT * FROM diary_entries WHERE user_id = ? LIMIT 1;
```

# Retrieving a specific row from a table

- Go to `app/Http/Controllers/DiaryEntryController.php`

## Query Builder

```
use Illuminate\Support\Facades\DB;
```

```
public function display_diary()
{
    $userId = Auth::id(); // Get the authenticated user's ID

    // Fetch all diary entries for the authenticated user
    $diaryEntries = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->first();

    return response()->json($diaryEntries);
}
```

Use the `table` method on the DB facade to specify the table name and then use the `first` method to a single record.

## Eloquent

```
public function display_diary()
{
    $userId = Auth::id();
    $diaryEntry = DiaryEntry::where('user_id', $userId)->first();

    return response()->json($diaryEntry);
}
```

The Eloquent model for the `diary_entries` table

# Retrieving specific rows from a table

```
SELECT date, content
FROM diary_entries
WHERE user_id = ? AND content LIKE '%day%';
```

```
public function display_diary()
{
    $userId = Auth::id();

    $results = DB::table('diary_entries')
        ->select('date', 'content')
        ->where('user_id', $userId)
        ->where('content', 'like', '%day%')
        ->get();

    return response()->json($results);
}
```



```
1 [
2   {
3     "date": "2024-08-01",
4     "content": "Had a productive day, finished a lot of tasks at work."
5   },
6   {
7     "date": "2024-08-03",
8     "content": "Feeling overwhelmed and a bit down after a tough day."
9   },
10  {
11    "date": "2024-08-08",
12    "content": "Had a rough day at work, feeling quite sad."
13  },
14  {
15    "date": "2024-08-09",
16    "content": "Productive day, but a bit of anxiety is creeping in."
17  },
18  {
19    "date": "2024-08-10",
20    "content": "Spent the day relaxing, feeling happy and content."
21  },
22  {
23    "date": "2024-08-10",
24    "content": "Spent the day relaxing at spa."
25  }
26 ]
```



## 3

# Retrieve the values of a single column

To retrieve the values of a single column (an Illuminate\Support\Collection instance), you may use the **pluck method**.

```
public function display_diary()
{
    $userId = Auth::id(); // Get the authenticated user's ID
    $contents = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->pluck('content');

    return response()->json($contents);
}
```



The values of a single column as a flat array or collection

```
1  [
2
3      "This is new",
4      "Very Happy Today"
5  ]
```

Or you can use **'select'**:

```
public function display_diary()
{
    $userId = Auth::id();
    $contents = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->select('content')
        ->get();

    return response()->json($contents);
}
```



Each result as an object with column names as properties.

```
1  [
2      {
3          "content": "This is new"
4      },
5      {
6          "content": "Very Happy Today"
7      }
8  ]
```

# Retrieve an associative array of **Key** and **Value**

You may specify the column that the resulting collection should use as its keys by providing a second argument to the pluck method

```
public function display_diary()
{
    $userId = Auth::id(); // Get the authenticated user's ID
    $contents = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->pluck('content', 'date');

    return response()->json($contents);
}
```



1	{	
2		"2024-08-23": "Very Happy Today"
3	}	

- If multiple records **share the same key, only the last one will be retained** because keys in associative arrays must be unique.

## 4

# Retrieve the values of multiple columns

```
public function display_diary()  
{  
    $userId = Auth::id();  
    $contents = DB::table('diary_entries')  
        ->where('user_id', $userId)  
        ->select('content', 'date')  
        ->get();  
  
    return response()->json($contents);  
}
```



```
1  [  
2      {  
3          "content": "This is new",  
4          "date": "2024-08-23"  
5      },  
6      {  
7          "content": "Very Happy Today",  
8          "date": "2024-08-23"  
9      }  
10 ]
```

# Query **distinct** results

```
public function display_diary()
{
    $userId = Auth::id();
    $contents = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->select('content')
        ->get();

    return response()->json($contents);
}
```



```
1 [
2   {
3     "content": "Happy makmak"
4   },
5   {
6     "content": "examination's day"
7   },
8   {
9     "content": "sad"
10  },
11  {
12    "content": "sad"
13  }
14 ]
```

```
public function display_diary()
{
    $userId = Auth::id();
    $contents = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->select('content')
        ->distinct()
        ->get();

    return response()->json($contents);
}
```



```
1 [
2   {
3     "content": "Happy makmak"
4   },
5   {
6     "content": "examination's day"
7   },
8   {
9     "content": "sad"
10  }
11 ]
```

# The orderBy Method

```
public function display_diary()  
{  
    $userId = Auth::id();  
    $contents = DB::table('diary_entries')  
        ->where('user_id', $userId)  
        ->select('content', 'date')  
        ->get();  
  
    return response()->json($contents);  
}
```



```
1 [
2   {
3     "content": "Happy makmak",
4     "date": "2024-09-04"
5   },
6   {
7     "content": "examination's day",
8     "date": "2024-09-04"
9   },
10  {
11    "content": "sad",
12    "date": "2024-09-02"
13  },
14  {
15    "content": "sad",
16    "date": "2024-09-01"
17  }
18 ]
```

```
public function display_diary()  
{  
    $userId = Auth::id();  
    $contents = DB::table('diary_entries')  
        ->where('user_id', $userId)  
        ->select('content', 'date')  
        ->orderBy('date', 'asc')  
        ->orderBy('content', 'desc')  
        ->get();  
  
    return response()->json($contents);  
}
```



```
1 [
2   {
3     "content": "sad",
4     "date": "2024-09-01"
5   },
6   {
7     "content": "sad",
8     "date": "2024-09-02"
9   },
10  {
11    "content": "Happy makmak",
12    "date": "2024-09-04"
13  },
14  {
15    "content": "examination's day",
16    "date": "2024-09-04"
17  }
18 ]
```

## 5

# Retrieving aggregate values

count, max, min, avg, sum

```
public function diary_count()
{
    $userId = Auth::id();

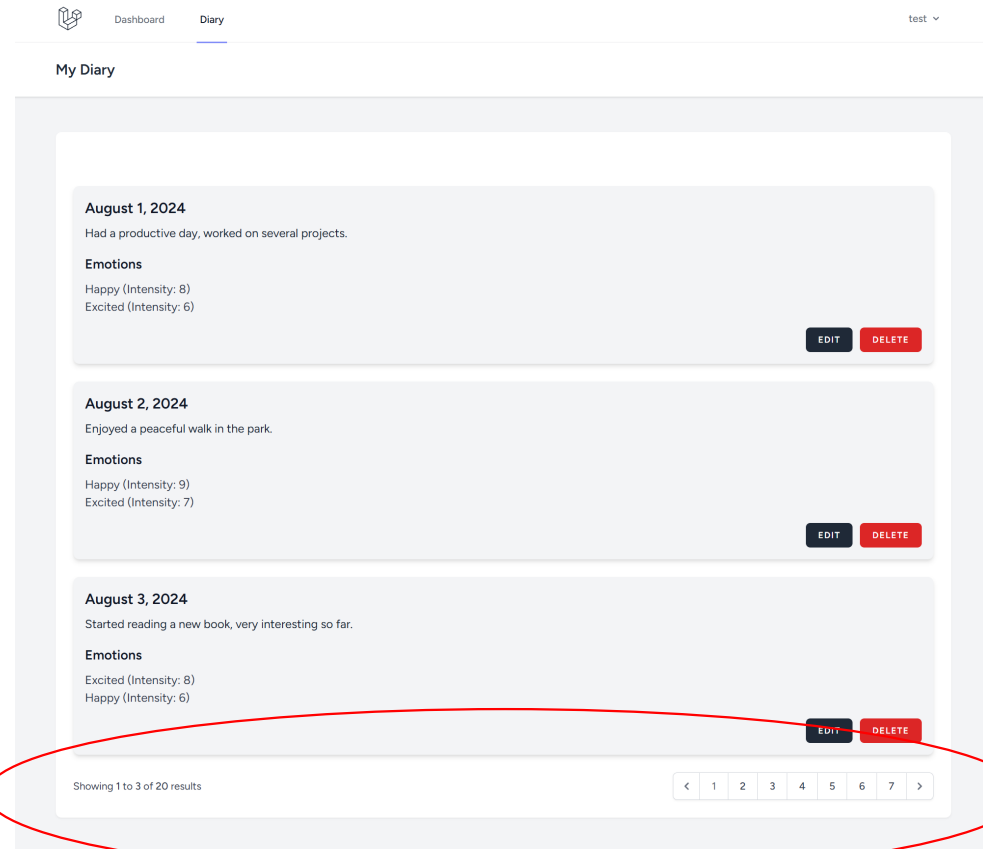
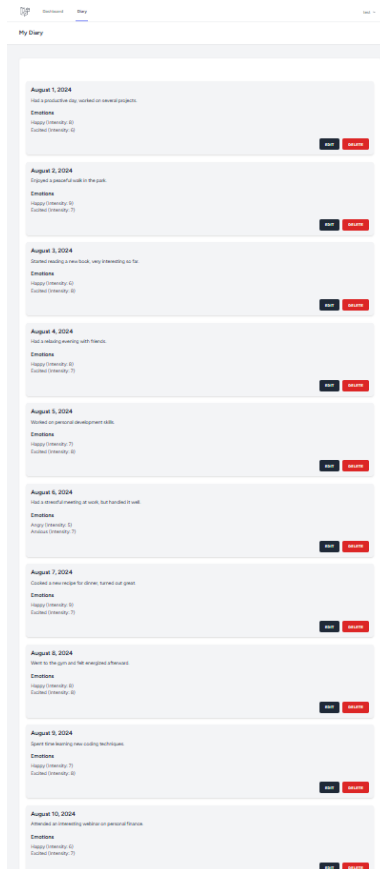
    $diary_count = DB::table('diary_entries')
        ->where('user_id', $userId)
        ->count();
    return response()->json(['diary_count' => $diary_count]);
}
```



1	{	"diary_count": 4
2		
3	}	

6

# Pagination in Laravel



# Raw Expressions

Laravel can not guarantee that any query using raw expressions is protected against SQL injection vulnerabilities.

- To create a raw string expression, you may use the **raw** method provided by the DB facade:

```
public function diary_count()
{
    $userId = Auth::id();

    $diary_count = DB::table('diary_entries')
        ->select('date', DB::raw('count(*) as count'))
        ->where('user_id', $userId)
        ->groupBy('date')
        ->having('count', '>=', 2)
        ->get();
    return response()->json(['diary_count' => $diary_count]);
}
```



```
1 {
2   "diary_count": [
3     {
4       "date": "2024-09-06",
5       "count": 3
6     },
7     {
8       "date": "2024-09-05",
9       "count": 2
10    }
11  ]
12 }
```



# Raw Expressions: example

Instead of using the `DB::raw` method, you may also use the following methods to insert a raw expression into various parts of your query.

- `selectRaw`
- `whereRaw`
- `havingRaw`
- `orderByRaw`
- `groupByRaw`

```
public function list_menus()  
{  
    $menus = DB::table('menus')  
        ->orderByRaw('name ASC, price ASC')  
        ->get();  
    return $menus;  
}
```

=

```
public function list_menus()  
{  
    $menus = DB::table('menus')  
        ->orderBy('name', 'ASC')  
        ->orderBy('price', 'ASC')  
        ->get();  
    return $menus;  
}
```

# Raw Expressions: example

- The **whereRaw** method can be used to inject a raw "where" clause into your query.
- It accept an optional **array of bindings** as their **second argument**.

```
$orders = DB::table('orders')  
    ->whereRaw('price > IF(shop = "main", ?, 100)', [200])  
    ->get();
```

Raw SQL clause: 'price > IF(shop = "main", ?, 100)'

Binding array: [200]

- If the column shop equals "main", then the ? placeholder is replaced with 200, which comes from the array of bindings. → **price > 200**
- If the shop column is not "main", then the value 100 is used. → **price > 100**

# Raw Expressions: example

- The **havingRaw** method can be used to inject a raw "having" clause into your query.
- It accept an optional **array of bindings** as their **second argument**.

```
public function diary_count()
{
    $userId = Auth::id();

    $diary_count = DB::table('diary_entries')
        ->select('date', DB::raw('count(*) as count'))
        ->where('user_id', $userId)
        ->groupBy('date')
        ->havingRaw('COUNT(*) > 2 AND COUNT(*) < 4')
        ->get();
    return response()->json(['diary_count' => $diary_count]);
}
```

```
SELECT date, COUNT(*) as count
FROM diary_entries
WHERE user_id = ?
GROUP BY date
HAVING COUNT(*) > 2 AND COUNT(*) < 4;
```

## 8

# Join

- To perform a basic "inner join", you may use the **join method** on a query builder instance.
- To perform a "left join" or "right join", use the **leftJoin** or **rightJoin** methods.

```
public function count_happy_diary()
{
    $userId = Auth::id();
    $happyEmotionCount = DB::table('users as u')
        ->join('diary_entries as de', 'u.id', '=', 'de.user_id')
        ->join('diary_entry_emotions as dee', 'de.id', '=', 'dee.diary_entry_id')
        ->where('u.id', $userId)
        ->where('dee.emotion_id', 1)
        ->count('de.id');

    return response()->json(['happyEmotionCount' => $happyEmotionCount]);
}
```

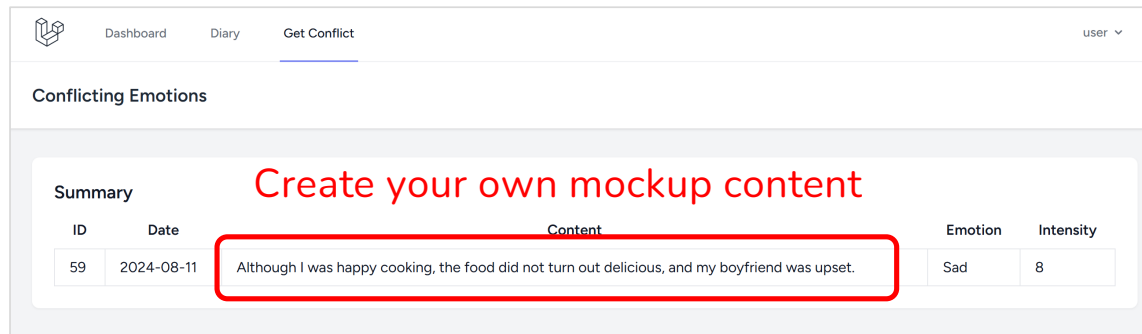
Example result

1	{	"happyEmotionCount": 7
2		
3	}	

# Your turn: Lab Assignment

## Building Basic Emotion Analysis Functionality in Laravel

- Identifying conflicting emotions: you have to develop a page that display all diary entries where the user expressed the emotion “Sad” (emotion\_id = 2) but mentioned the word “happy” in the content of the diary entry.
- Instructions:
  - 1) Write the raw SQL query that accomplishes the task
  - 2) Implement the function in the controller
  - 3) Define Routes
  - 4) Create Views



Dashboard Diary Get Conflict user ▾

Conflicting Emotions

Summary

Create your own mockup content

ID	Date	Content	Emotion	Intensity
59	2024-08-11	Although I was happy cooking, the food did not turn out delicious, and my boyfriend was upset.	Sad	8

# Self practice (optional)

## Quiz Prompt:

Design a **search feature** for a diary app that lets users find entries based on a specific emotion and a keyword. For example, allow users to search for entries where they felt "happy" and mentioned "vacation." This feature enhances the app by helping users reflect on their emotional experiences and discover patterns in their diary.