# Basic Database Usage

DB::connection('connection\_name');

## // Running A General Statement

DB::statement('drop table users');

## // Listening For Query Events

DB::listen(function($sql, $bindings, $time){ code\_here; });

## // Pessimistic Locking

DB::table('users')->where('votes', '>', 100)->sharedLock()->get();

DB::table('users')->where('votes', '>', 100)->lockForUpdate()->get();

## // Database Transactions

DB::transaction(function()

{

DB::table('users')->update(['votes' => 1]);

DB::table('posts')->delete();

});

DB::beginTransaction();

DB::rollBack();

DB::commit();

# DML

## // Inserts

DB::table('users')->insert(

['email' => 'john@example.com', 'votes' => 0]

);

$id = DB::table('users')->insertGetId(

['email' => 'john@example.com', 'votes' => 0]

);

DB::table('users')->insert([

['email' => 'taylor@example.com', 'votes' => 0],

['email' => 'dayle@example.com', 'votes' => 0]

]);

## // Updates

DB::table('users') ->where('id', 1) ->update(['votes' => 1]);

DB::table('users')->increment('votes');

DB::table('users')->increment('votes', 5);

DB::table('users')->decrement('votes');

DB::table('users')->decrement('votes', 5);

DB::table('users')->increment('votes', 1, ['name' => 'John']);

## // Deletes

DB::table('users')->where('votes', '<', 100)->delete();

DB::table('users')->delete();

DB::table('users')->truncate();

# Query

## // Running A Select Query

$results = DB::select('select \* from users where id = ?', [1]);

$results = DB::select('select \* from users where id = :id', ['id' => 1]);

# Query Builder

## // Retrieving All Rows From A Table

DB::table('name')->get();

## // Chunking Results From A Table

DB::table('users')->chunk(100, function($users)

{

foreach ($users as $user)

{ }

});

## // Retrieving A Single Row From A Table

$user = DB::table('users')->where('name', 'John')->first();

DB::table('name')->first();

## // Retrieving A Single Column From A Row

$name = DB::table('users')->where('name', 'John')->pluck('name');

DB::table('name')->pluck('column');

## // Retrieving A List Of Column Values

$roles = DB::table('roles')->lists('title');

$roles = DB::table('roles')->lists('title', 'name');

## // Specifying A Select Clause

$users = DB::table('users')->select('name', 'email')->get();

$users = DB::table('users')->distinct()->get();

$users = DB::table('users')->select('name as user\_name')->get();

## // Adding A Select Clause To An Existing Query

$query = DB::table('users')->select('name');

$users = $query->addSelect('age')->get();

## // Using Where Operators

$users = DB::table('users')->where('votes', '>', 100)->get();

$users = DB::table('users') ->where('votes', '>', 100) ->orWhere('name', 'John') ->get();

$users = DB::table('users') ->whereBetween('votes', [1, 100])->get();

$users = DB::table('users') ->whereNotBetween('votes', [1, 100])->get();

$users = DB::table('users') ->whereIn('id', [1, 2, 3])->get();

$users = DB::table('users') ->whereNotIn('id', [1, 2, 3])->get();

$users = DB::table('users') ->whereNull('updated\_at')->get();

DB::table('name')->whereNotNull('column')->get();

## // Dynamic Where Clauses

$admin = DB::table('users')->whereId(1)->first();

$john = DB::table('users') ->whereIdAndEmail(2, 'john@doe.com') ->first();

$jane = DB::table('users') ->whereNameOrAge('Jane', 22) ->first();

## // Order By, Group By, And Having

$users = DB::table('users') ->orderBy('name', 'desc') ->groupBy('count')

->having('count', '>', 100)->get();

DB::table('name')->orderBy('column')->get();

DB::table('name')->orderBy('column','desc')->get();

DB::table('name')->having('count', '>', 100)->get();

## // Offset & Limit

$users = DB::table('users')->skip(10)->take(5)->get();

## Joins

### // Basic Join Statement

DB::table('users')

->join('contacts', 'users.id', '=', 'contacts.user\_id')

->join('orders', 'users.id', '=', 'orders.user\_id')

->select('users.id', 'contacts.phone', 'orders.price')

->get();

### // Left Join Statement

DB::table('users')

->leftJoin('posts', 'users.id', '=', 'posts.user\_id')

->get();

select \* from users where name = 'John' or (votes > 100 and title <> 'Admin')

DB::table('users')

->where('name', '=', 'John')

->orWhere(function($query)

{

$query->where('votes', '>', 100)

->where('title', '<>', 'Admin');

})

->get();

## Aggregates

$users = DB::table('users')->count();

$price = DB::table('orders')->max('price');

$price = DB::table('orders')->min('price');

$price = DB::table('orders')->avg('price');

$total = DB::table('users')->sum('votes');

## Raw Expressions

$users = DB::table('users')

->select(DB::raw('count(\*) as user\_count, status'))

->where('status', '<>', 1) ->groupBy('status') ->get();

## // return rows

DB::select('select \* from users where id = ?', array('value'));

## // return nr affected rows

DB::insert('insert into foo set bar=2');

DB::update('update foo set bar=2');

DB::delete('delete from bar');

## // returns void

DB::statement('update foo set bar=2');

## // raw expression inside a statement

DB::table('name')->select(DB::raw('count(\*) as count, column2'))->get();

## // Unions

// The unionAll() method is also available, and has the same method signature as union.

$first = DB::table('users')->whereNull('first\_name');

$users = DB::table('users')->whereNull('last\_name')->union($first)->get();

Contents

[Basic Database Usage 1](#_Toc511913720)

[Running A General Statement 1](#_Toc511913721)

[Listening For Query Events 1](#_Toc511913722)

[Database Transactions 1](#_Toc511913723)

[DML 2](#_Toc511913724)

[Inserts 2](#_Toc511913725)

[Updates 2](#_Toc511913726)

[Deletes 2](#_Toc511913727)

[Query 3](#_Toc511913728)

[Running A Select Query 3](#_Toc511913729)

[Query Builder 3](#_Toc511913730)

[Retrieving All Rows From A Table 3](#_Toc511913731)

[Chunking Results From A Table 3](#_Toc511913732)

[Retrieving A Single Row From A Table 3](#_Toc511913733)

[Retrieving A Single Column From A Row 3](#_Toc511913734)

[Retrieving A List Of Column Values 3](#_Toc511913735)

[Specifying A Select Clause 3](#_Toc511913736)

[Adding A Select Clause To An Existing Query 3](#_Toc511913737)

[Using Where Operators 4](#_Toc511913738)

[Dynamic Where Clauses 4](#_Toc511913739)

[Order By, Group By, And Having 4](#_Toc511913740)

[Offset & Limit 4](#_Toc511913741)

[Joins 5](#_Toc511913742)

[Basic Join Statement 5](#_Toc511913743)

[Left Join Statement 5](#_Toc511913744)

[Aggregates 5](#_Toc511913745)

[Raw Expressions 6](#_Toc511913746)

[return rows 6](#_Toc511913747)

[return nr affected rows 6](#_Toc511913748)

[returns void 6](#_Toc511913749)

[raw expression inside a statement 6](#_Toc511913750)

[Unions 6](#_Toc511913751)

[Pessimistic Locking 6](#_Toc511913752)