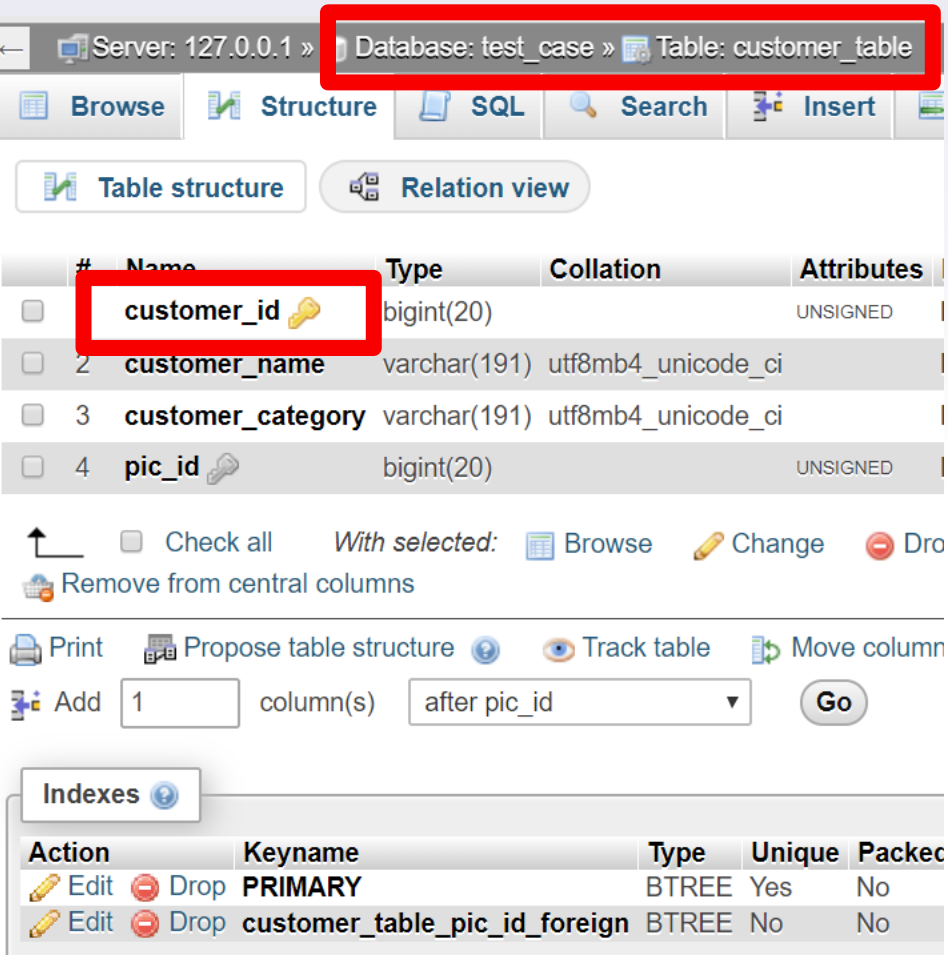
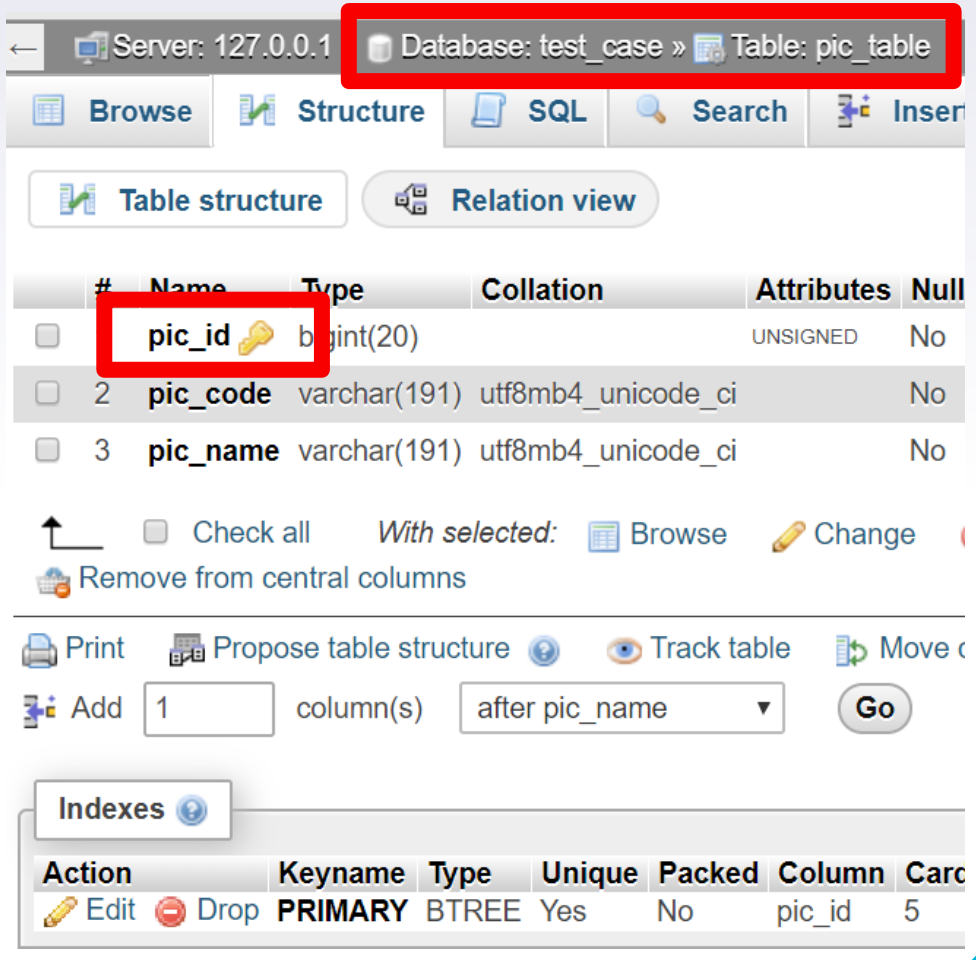
1

OPEN CONNECTION

Connect Laravel project with XAMPP


Server: 127.0.0.1 » Database: test_case » Table: customer_table
Browse
Structure
SQL
Search
Insert
Table structure
Relation view
Name Type Collation Attributes
customer_id bigint(20) UNSIGNED
2 customer_name varchar(191) utf8mb4_unicode_ci
3 customer_category varchar(191) utf8mb4_unicode_ci
4 pic_id bigint(20) UNSIGNED
Check all With selected: Browse Change
Remove from central columns
Print Propose table structure Track table
Add 1 column(s) after pic_id Go
Indexes
Action Keyname Type Unique Packed
Edit Drop PRIMARY BTREE Yes No
Edit Drop customer_table_pic_id_foreign BTREE No No


Server: 127.0.0.1 Database: test_case » Table: pic_table
Browse
Structure
SQL
Search
Table structure
Relation view
Name Type Collation Attributes Null
pic_id bigint(20) UNSIGNED No
2 pic_code varchar(191) utf8mb4_unicode_ci No
3 pic_name varchar(191) utf8mb4_unicode_ci No
Check all With selected: Browse Change
Remove from central columns
Print Propose table structure Track table
Add 1 column(s) after pic_name Go
Indexes
Action Keyname Type Unique Packed Column
Edit Drop PRIMARY BTREE Yes No pic_id 5

define database

related file(s): **.env**

required command: **php artisan serve**

```
DB_DATABASE=test_case
DB_USERNAME=root
DB_PASSWORD=
```

```
E:\COMP6114 - WEB PROGRAMMING (Laravel Framework)\BCA Project\empty-project>php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
```

2

MODEL

Create relational database

create model

when you run this command, Laravel **automatically** create its migration with format **year_month_date_hhmmss_create_#model.name[+s]_table.php** rename this will cause **error**, it might be decided by Laravel to add auto generated timestamps in each migration name to ensure it'll successfully **sequencially**

related file(s): **providers > Customer.php**
providers > Pic.php

required command: **php artisan make:model #model.name -m**

```php
class Pic extends Model
{
    protected $table = 'pic_table';
    public $timestamps = false;
    protected $primaryKey = 'pic_id';

    public function pic_func() {
        return $this->hasMany( related: Customer
    }
}
```

```php
class Customer extends Model
{
    use Sortable;
    public $sortable = ['customer_name', 'customer_category', 'pic_id'];

    protected $table = 'customer_table';
    public $timestamps = false;
    protected $primaryKey = 'customer_id';

    public function customer_func() {
        return $this->belongsTo( related: Pic::class, foreignKey: 'pic_id');
    }
}
```

for **sorting** feature

create model

when you run this command, Laravel **automatically** create its migration with format **year_month_date_hhmmss_create_#model.name[+s]_table.php** rename this will cause **error**, it might be decided by Laravel to add auto generated timestamps in each migration name to ensure it'll successfully **sequencially**

related file(s): **providers > Customer.php**
providers > Pic.php

required command: **php artisan make:model #model.name -m**

```php
class Pic extends Model
{
    protected $table = 'pic_table';
    public $timestamps = false;
    protected $primaryKey = 'pic_id';

    public function pic_func() {
        return $this->hasMany( related: Customer::class, foreignKey: 'pic_id');
    }
}
```

```php
class Customer extends Model
{
    use Sortable;
    public $sortable = ['customer_name', 'customer_category', 'pic_id'];

    protected $table = 'customer_table';
    public $timestamps = false;
    protected $primaryKey = 'customer_id';

    public function customer_func() {
        return $this->belongsTo( related: Pic::class, foreignKey: 'pic_id');
    }
}
```

create migration

related file(s): **migrations > …_create_pics_table.php**
migrations > …_create_customers_table.php
required command: **php artisan make:model #model.name -m**

```php
Schema::create( table: 'pic_table', function (Blueprint $table) {
    $table->bigIncrements( column: 'pic_id');
    $table->string( column: 'pic_code');
    $table->string( column: 'pic_name');
});
```

```php
Schema::create( table: 'customer_table', function (Blueprint $table) {
    $table->bigIncrements( column: 'customer_id');
    $table->string( column: 'customer_name');
    $table->string( column: 'customer_category');
    $table->bigInteger( column: 'pic_id')->unsigned();
    $table->foreign( columns: 'pic_id')->references( columns: 'pic_id')->on( table: 'pic_table')->onDelete( action: 'cascade');
});
```

data types defined here **must be the same** with data types defined in MySQL

do migration

related file(s): #
required command: **php artisan migrate:fresh**

```
F:\COMP6114 - WEB PROGRAMMING (Laravel Framework)\BCA Project\empty-project>php artisan migrate:fresh
Dropped all tables successfully.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2020_01_23_052858_create_pics_table
Migrated:  2020_01_23_052858_create_pics_table
Migrating: 2020_01_23_053029_create_customers_table
Migrated:  2020_01_23_053029_create_customers_table
```

model with **foreign keys** must be migrated **after** its referenced model migrated

seed rows in tables

related file(s): **seeds > customer.php**
seeds > pic.php
required command: **php artisan db:seed**

```php
DB::table( table: 'pic_table')->insert([
    [
        'pic_code' => 'BUD',
        'pic_name' => 'BUDI',
    ],
    [
        'pic_code' => 'JOH',
        'pic_name' => 'JOHNATAM',
    ],
    [
        'pic_code' => 'STE',
        'pic_name' => 'STEVEN',
    ],
```

```php
DB::table( table: 'customer_table')->insert([
    [
        'customer_name' => 'Jonathan',
        'customer_category' => 'Debitur',
        'pic_id' => '1',
    ],
    [
        'customer_name' => 'Angeline',
        'customer_category' => 'Debitur',
        'pic_id' => '3',
    ],
    [
        'customer_name' => 'Aaron',
        'customer_category' => 'Nasabah',
        'pic_id' => '2',
    ],
```

attributes defined here **must be the same** with attributes defined in MySQL

3

CONTROLLER

Programming logic placement

define route

related file(s): **web.php**
required command: **#**

route action format **controller_name@function_name**

```php
<?php
use Illuminate\Support\Facades\Route;

Route::get( uri: '/', action: 'display@showData');
Route::get( uri: '/add', function () {return view( view: 'addview');})->name( name: 'add'); // alias route
Route::post( uri: '/add', action: 'add_controller@add_customer_data');
Route::post( uri: '/edit', action: 'edit_controller@edit')->name( name: 'edit_data');
Route::post( uri: '/delete', action: 'delete_controller@delete_data')->name( name: 'delete_data');
Route::get( uri: '/search', action: 'Select2Controller@loadData');
```

for **searching** feature

display tables in main page

related file(s): **display.php**
required command: **php artisan make:controller #controller.name**

```
function showData(){
    $cust = Customer::sortable()->paginate(3);
    return view( view: 'welcome', ['for_each_cust' => $cust], compact($cust));
}
```

for **sorting** feature

php syntax to **passing array of data** (in this case, it comes from Customer model)

add data

related file(s): **add_controller.php**
required command: **php artisan make:controller #controller.name**

```
$req->validate([
    'name'=>'required|string|max:100',
    'category'=>'required|in:Nasabah,Debitur',
    'Pic'=>'required',
]);
```

```
$new = new Customer;
$new->customer_name = $req->input( key: 'name');
$new->customer_category = $req->input( key: 'category');
$new->pic_id = $req->input( key: 'Pic');
$new->save();

return redirect( to: '/');
```

add data

related file(s): **add_controller.php**
required command: **php artisan make:controller #controller.name**

```
$req->validate([
    'name'=>'required|string|max:100',
    'category
    'Pic'=>'r
]);
```

```
$new = new Customer;
$new->customer_name = $req->input( key: 'name');
$new->customer_category = $req->input( key: 'category');
$new->pic_id = $req->input( key: 'Pic');
$new->save();


return redirect( to: '/');
```

edit data

related file(s): **edit_controller.php**
required command: **php artisan make:controller #controller.name**

```php
$data = $req->except( keys: '_token', 'category', 'Pic');
$kat = $req->input( key: 'category');
$pic = $req->input( key: 'Pic');

if(empty($data)) { }
else {
    foreach($data as $array)
    foreach($array as $id){
        if(is_nan((int)$id)) { }
        else {
            $item = Customer::findorfail($id);
            if(!empty($kat))
            $item->customer_category = $kat;

            if(!empty($pic))
            $item->pic_id = $pic;

            $item->save();
        }
    }
}
return redirect( to: '/');
```

edit data

related file(s): **edit_controller.php**
required command: **php artisan make:controller #controller.name**

```php
$data = $req->except( keys: '_token', 'category'
$kat = $req->input( key: 'category');
$pic = $req->input( key: 'Pic');

if(empty($data)) { }
else {
    foreach($data as $array)
    foreach($array as $id){
        if(is_nan((int)$id)) { }
        else {
            $item = Customer::findorfail($id);
            if(!empty($kat))
            $item->customer_category = $kat;

            if(!empty($pic))
            $item->pic_id = $pic;

            $item->save();
        }
    }
}
return redirect( to: '/');
```

delete data

related file(s): **delete_controller.php**
required command: **php artisan make:controller #controller.name**

```php
$data = $req->except( keys: '_token', 'category', 'Pic');

if (empty($data)) { }
else {
        foreach ($data as $id) {
            if (is_nan((int)$id)) { }
            else Customer::destroy($id);
        }

}
return redirect( to: '/');
```

search

related file(s): **Select2Controller.php**
required command: **php artisan make:controller #controller.name**

```php
if ($request->has( key: 'q')) {
    $search = $request->q;
    $data = DB::table( table: 'customer_table')
        ->leftJoin( table: 'pic_table', first: 'customer_table.pic_id', operator: '=', second: 'pic_table.pic_id')
        ->select( columns: 'customer_id', 'customer_name')
        ->where( column: 'customer_name', operator: 'LIKE', value: '%'.$search.'%')
        ->orWhere( column: 'customer_category', operator: 'LIKE', value: '%'.$search.'%')
        ->orWhere( column: 'pic_code', operator: 'LIKE', value: '%'.$search.'%')->get();
    return response()->json($data);
}
```

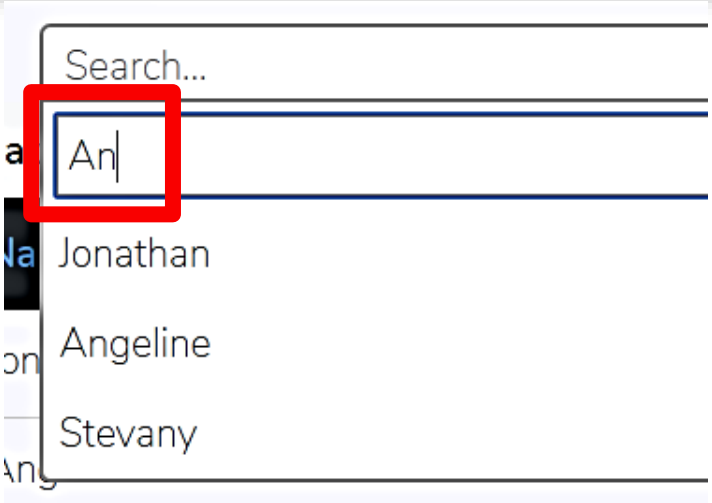
The **LEFT OUTER JOIN** (or simply LEFT JOIN) returns all records from the left table (customer_table), even if there are no matches in the right table (pic_table). But in this case, **JOIN** (or NATURAL JOIN) will return the same result as LEFT JOIN because it's impossible for customer to have no PIC (pic_id != NULL).

search

```
return response()->json($data);
```

127.0.0.1:8000/search?q=An

[{"customer_id":1,"customer_name":"Jonathan"},{"customer_id":2,"customer_name":"Angeline"},{"customer_id":5,"customer_name":"Stevany"}]



4

VIEW

UI/UX design

add view

related file(s): **addview.blade.php**
required command: **#**

```
<h3>ADD DATA</h3>
<form method="post" action="">
    {{ csrf_field() }} {{--performed on behalf of an authenticated user--}}
    <div class="">
        <div class="col-md-12 m-auto">
            <input type="text" name="name" placeholder="Tulis nama lengkap Anda.. (maks: 100 karakter)" class="fo
            @error('name')
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $message }}</strong>
                </span>
            @enderror
        </div>
```

```
$req->validate([
    'name'=> required|string|max:100',
    'category'=>'required|in:Nasabah,Debitur',
    'Pic'=>'required',
]);
```

must have a **same name**
with name defined in **add_controller.php**

welcome view (main page)

related file(s): **welcome.blade.php**
required command: **#**

```
<form id="formfield" method="post" action="" class="">
    {{csrf_field()}}
    <button type="button" class="btn btn-light font-weight-bold" value="Add Data" onclick="window.location.href = '{{route('add')}}'">Add Data</button>
    <button type="button" class="btn btn-light font-weight-bold" value="Edit Data" onclick="display_edit()">Edit Data</button>
    <button class="btn btn-light font-weight-bold" value="Delete Data" type="submit"
            onclick="document.getElementById('formfield').action = '{{route('delete_data')}}'">Delete</button>

    <table class="table table-responsive table-hover" style="...">
        <thead class="thead-dark text-center">
            <tr>
                <th scope="col" style="..."><input type="checkbox" id="check_head" onclick="checkAll()"></th>
                <th scope="col">@sortablelink('customer_name', 'Name')</th>
                <th scope="col">@sortablelink('pic_id', 'PIC')</th>
                <th scope="col">@sortablelink('customer_category', 'Category')</th>
            </tr>
        </thead>
```

function in **Javascript**

for **sorting** feature

welcome view (main page)

related file(s): **welcome.blade.php**
required command: **#**

```
<tbody>
    @foreach($for_each_cust as $a)
        <tr>
            <td><input type="checkbox" name="dataset[]" value="{{$a->customer_id}}" class="text-center"></td>
            <td>{{$a->customer_name}}</td>
            <td>{{$a->customer_func->pic_code}}</td>
            <td>{{$a->customer_category}}</td>
        </tr>
    @endforeach
</tbody>
```

javascript

related file(s): **welcome.blade.php**
required command: **#**

```
<script>
    function display_edit() {
        var a = document.getElementById("edit");

        if(a.style.display === "block") a.style.display = "none";
        else a.style.display = "block";
    }
```

javascript

related file(s): **welcome.blade.php**
required command: **#**

```
function checkAll(){
    var headCheckbox = document.getElementById('check_head');
    for(let inp of document.querySelectorAll('input')){
        if(inp.type === "checkbox" && inp.name === "dataset[]")
            !headCheckbox.checked ? inp.checked = false : inp.checked = true;
    }
}
```

ajax for search

related file(s): **welcome.blade.php**
required command: **#**

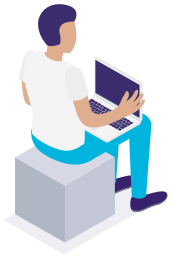
```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.3/js/select2.min.js"></script>
<script type="text/javascript">
    $('.search').select2({
        placeholder: 'Search...',
        ajax: {
            url: '/search',
            dataType: 'json',
            delay: 250,
            processResults: function (data) {
                console.log(data);
                return {
                    results:  $.map(data, function (item) {
                        return { text: item.customer_name }
                    })
                };
```

ajax stands for **a**synchronous **j**avascript and **x**ml, used to be a built-in object to **request data** from web server

json stands for **j**avascript **o**bject **n**otation, used to be an interchange data format

THANKYOU

Any questions?



You can find me at:

- nicholas.dominic@binus.ac.id
- linkedin.com/in/nicholas-dominic