

**LAPORAN PRAKTIKUM  
PERANCANGAN DAN PEMROGRAMAN WEB**

**MODUL 12  
MIGRATION**



**Universitas  
Telkom**

Oleh:

Aulia Jasifa Br Ginting      2311104060

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK  
DIREKTORAT KAMPUS PURWOKERTO  
UNIVERSITAS TELKOM  
2025**

## **BAB I**

### **PENDAHULUAN**

#### **A. Dasar Teori**

Migration pada framework Laravel merupakan mekanisme pengelolaan struktur database berbasis kode yang berfungsi sebagai version control untuk database. Dengan migration, pengembang dapat membuat, mengubah, dan menghapus tabel maupun kolom database secara terstruktur tanpa harus langsung berinteraksi dengan database secara manual. Fitur ini sangat membantu dalam menjaga konsistensi struktur database antar lingkungan pengembangan, serta memungkinkan perubahan skema database dilakukan tanpa menghilangkan data yang sudah ada. Selain itu, migration mendukung proses rollback sehingga struktur database dapat dikembalikan ke kondisi sebelumnya apabila terjadi kesalahan, yang menjadikan pengembangan aplikasi lebih aman dan terkontrol.

Model dalam Laravel berperan sebagai representasi dari tabel database yang digunakan untuk mengelola data menggunakan konsep CRUD (Create, Read, Update, Delete). Model membantu memisahkan logika bisnis dari logika tampilan dan database sehingga kode menjadi lebih rapi, terstruktur, dan mudah dipelihara. Laravel menyediakan beberapa cara untuk mengakses database melalui model, yaitu DB Facade (Raw SQL), Query Builder, dan Eloquent ORM. Raw SQL digunakan untuk menjalankan perintah SQL secara langsung dan biasanya dipakai untuk query yang kompleks. Query Builder menyediakan cara yang lebih aman dan fleksibel dalam menyusun query SQL menggunakan metode-metode bawaan Laravel. Sementara itu, Eloquent ORM memungkinkan pengembang berinteraksi dengan database menggunakan pendekatan berbasis objek, sehingga setiap baris data direpresentasikan sebagai objek, yang membuat proses pengembangan aplikasi menjadi lebih cepat, efisien, dan aman dari serangan seperti SQL Injection.

#### **B. Tujuan**

1. Memahami fungsi dan cara kerja model pada framework laravel
2. Mampu menerapkan migration pada aplikasi

## BAB II

### HASIL

#### A. GUIDED (Praktikum Terbimbing)

##### 1. File Konfigurasi

The screenshot shows the Visual Studio Code interface with the .env file open. The .env file contains environment variables for a MySQL database connection:

```
BCRYPT_ROUNDS=12
LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATED_CHANNEL=null
LOG_LEVEL=debug
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=2311104060_ppw_jasifa
DB_USERNAME=root
DB_PASSWORD=
```

The screenshot shows the Visual Studio Code interface with the database.php file open in the config directory. The file defines a database connection for MySQL:

```
'mysql' => [
    'driver' => 'mysql',
    'url' => env('DB_URL'),
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', '2311104060_ppw_jasifa'),
    'username' => env('DB_USERNAME', 'root'),
    'password' => env('DB_PASSWORD', ''),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => env('DB_CHARSET', 'utf8mb4'),
    'collation' => env('DB_COLLATION', 'utf8mb4_unicode_ci'),
    'prefix' => '',
    'prefix_indexes' => true,
    'strict' => true,
    'engine' => null,
    'options' => extension_loaded('pdo_mysql') ? array_filter([
        PHP_VERSION_ID >= 80500 ? \Pdo\MySQL::ATTR_SSL_CA : \PDO::MYSQL_ATTR_SSL_CA
    ]) : []
],
```

##### 2. File migration bawaan laravel

The screenshot shows a file explorer window displaying the migrations directory. It lists several migration files with their names, dates modified, types, and sizes:

Name	Date modified	Type	Size
0001_01_000000_create_users_table	11/25/2025 2:16 PM	PHP Source File	2 KB
0001_01_000001_create_cache_table	11/25/2025 2:16 PM	PHP Source File	1 KB
0001_01_000002_create_jobs_table	11/25/2025 2:16 PM	PHP Source File	2 KB
2025_12_18_032441_create_mahasiswa_table	12/18/2025 10:46 AM	PHP Source File	1 KB
2025_12_18_040923_alter_mahasiswa_table	12/18/2025 11:11 AM	PHP Source File	1 KB

##### 3. Migration Rollback

The screenshot shows a terminal window running the command `php artisan migrate:rollback --step=1`. The output indicates that the migration step 1 is being rolled back.

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan> php artisan migrate:rollback --step=1
INFO Rolling back migrations.
2025_12_18_032441_create_mahasiswa_table ..... 23.32ms DONE
```

# LATIHAN

## 1. Membuat Migration

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan> php artisan make:migration create_mahasiswa_table --create=mahasiswa
[INFO] Migration [C:\xampp\htdocs\Laravel_Praktikum\latihan\database\migrations\2025_12_18_032441_create_mahasiswa_table.php] created successfully.
```

```
database > migrations > 2025_12_18_032441_create_mahasiswa_table.php
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     public function up(): void
10    {
11        Schema::create('mahasiswa', function (Blueprint $table) {
12            $table->id();
13            $table->char('nim', 8);
14            $table->string('nama');
15            $table->string('tempat_lahir');
16            $table->date('tanggal_lahir');
17            $table->string('fakultas');
18            $table->string('jurusan');
19            $table->decimal('ipk', 3, 2);
20            $table->timestamps();
21        });
22    }
23    public function down()
24    {
25        Schema::dropIfExists('mahasiswa');
26    }
27};
```

The screenshot shows the 'mahasiswa' table structure in phpMyAdmin. The table has 10 columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	bigint(20)	UNSERIALIZED	No	None	AUTO_INCREMENT			Change  Drop  More
2	<code>nim</code>	char(8)	utf8mb4_unicode_ci	No	None				Change  Drop  More
3	<code>nama</code>	varchar(255)	utf8mb4_unicode_ci	No	None				Change  Drop  More
4	<code>tempat_lahir</code>	varchar(255)	utf8mb4_unicode_ci	No	None				Change  Drop  More
5	<code>tanggal_lahir</code>	date		No	None				Change  Drop  More
6	<code>fakultas</code>	varchar(255)	utf8mb4_unicode_ci	No	None				Change  Drop  More
7	<code>jurusan</code>	varchar(255)	utf8mb4_unicode_ci	No	None				Change  Drop  More
8	<code>created_at</code>	timestamp		Yes	NULL				Change  Drop  More
9	<code>updated_at</code>	timestamp		Yes	NULL				Change  Drop  More
10	<code>ipk</code>	decimal(3,2)		No	1.00				Change  Drop  More

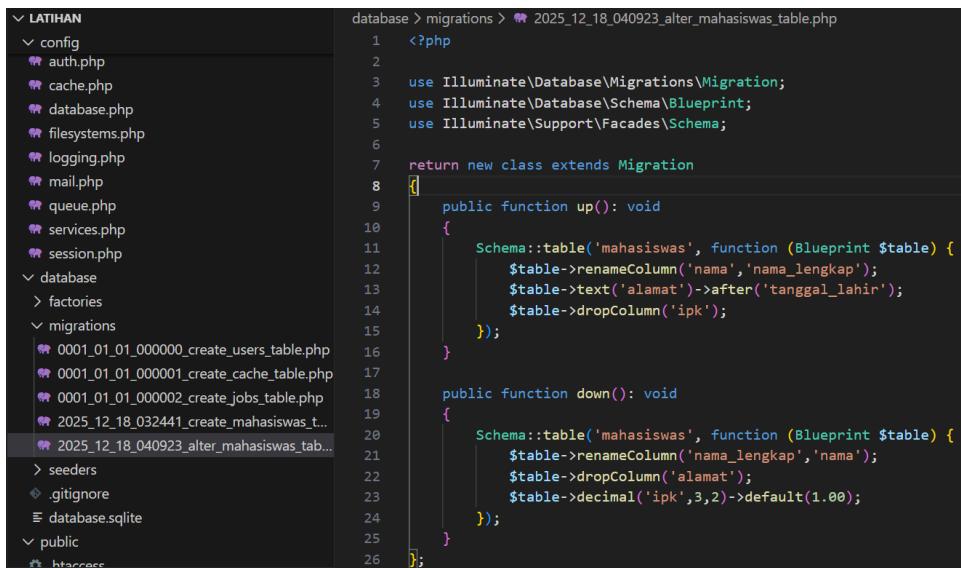
## 2. Modifikasi file migration

```
database > migrations > 2025_12_18_032441_create_mahasiswa_table.php
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     public function up(): void
10    {
11        Schema::create('mahasiswa', function (Blueprint $table) {
12            $table->id();
13            $table->char('nim', 8)->unique();
14            $table->string('nama');
15            $table->string('tempat_lahir');
16            $table->date('tanggal_lahir');
17            $table->string('fakultas');
18            $table->string('jurusan');
19            $table->decimal('ipk', 3, 2)->default(1.00);
20            $table->timestamps();
21        });
22    }
23    public function down()
24    {
25        Schema::dropIfExists('mahasiswa');
26    }
27};
```

### 3. Alter table migration

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan> composer require doctrine/dbal
./composer.json has been updated
Running composer update doctrine/dbal
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
- Locking doctrine/dbal (4.4.1)
- Locking doctrine/deprecations (1.1.5)
- Locking psr/cache (3.0.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
- Downloading psr/cache (3.0.0)
- Downloading doctrine/deprecations (1.1.5)
- Downloading doctrine/dbal (4.4.1)
- Installing psr/cache (3.0.0): Extracting archive
- Installing doctrine/deprecations (1.1.5): Extracting archive
- Installing doctrine/dbal (4.4.1): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
```

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan> php artisan make:migration alter_mahasiswa_table --table=mahasiswa
INFO Migration [C:\xampp\htdocs\Laravel_Praktikum\latihan\database\migrations\2025_12_18_040923_alter_mahasiswa_table.php] created successfully.
```



The screenshot shows a code editor with two panes. The left pane displays a file tree for a Laravel project named 'LATIHAN'. The right pane shows the content of the file '2025\_12\_18\_040923\_alter\_mahasiswa\_table.php'.

```
database > migrations > 2025_12_18_040923_alter_mahasiswa_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      public function up(): void
10     {
11         Schema::table('mahasiswa', function (Blueprint $table) {
12             $table->renameColumn('nama', 'nama_lengkap');
13             $table->text('alamat')->after('tanggal_lahir');
14             $table->dropColumn('ipk');
15         });
16     }
17
18     public function down(): void
19     {
20         Schema::table('mahasiswa', function (Blueprint $table) {
21             $table->renameColumn('nama_lengkap', 'nama');
22             $table->dropColumn('alamat');
23             $table->decimal('ipk',3,2)->default(1.00);
24         });
25     }
26 };
```

## MODUL 12.2 (DB Facade , Eloquent ORM, Query Builder)

### 1. Input Data menggunakan raw SQL Queries

```
routes > 🐾 web.php
1   <?php
2
3   use Illuminate\Support\Facades\Route;
4   use App\Http\Controllers\MahasiswaController;
5
6   Route::get('/insert-data', [MahasiswaController::class, 'insertdata']);
7   Route::get('/select-data', [MahasiswaController::class, 'selectData']);
8   Route::get('/update-data', [MahasiswaController::class, 'updateData']);
9   Route::get('/delete-data', [MahasiswaController::class, 'deleteData']);
```

```
app > Http > Controllers > 🐾 MahasiswaController.php
1   <?php
2   namespace App\Http\Controllers; use Illuminate\Http\Request;
3
4   use Illuminate\Support\Facades\DB;
5
6   class MahasiswaController extends Controller
7   {
8       public function insertData()
9       {
10           $result = DB::insert("")
11               INSERT INTO mahasiswa
12                   (nim, nama_lengkap, tempat_lahir, tanggal_lahir, fakultas, jurusan, alamat)
13                   VALUES
14                   ('20104064', 'Muhammad Nur Hamada', 'Purwokerto', '2002-02-02', 'Teknik', 'Informatika',
15                   ");
16
17           dump($result);
18       }
19   }
```

← ⏪ ⓘ 127.0.0.1:8000/insert-data

true // app\Http\Controllers\MahasiswaController.php:19

### 2. Input Data menggunakan Query Builder

```
app > Http > Controllers > MahasiswaController.php
1  <?php
2  namespace App\Http\Controllers; use Illuminate\Http\Request;
3
4  use Illuminate\Support\Facades\DB;
5
6  class MahasiswaController extends Controller
7  {
8      public function insertData(){
9          $mahasiswa = new Mahasiswa;
10         $mahasiswa->nim = '19003036';
11         $mahasiswa->nama_lengkap = 'Muhammad Hamada';
12         $mahasiswa->tempat_lahir = 'Bandung';
13         $mahasiswa->tanggal_lahir = '2002-02-02';
14         $mahasiswa->alamat = 'Jl. Merdeka No. 10';
15         $mahasiswa->fakultas = 'Teknik';
16         $mahasiswa->jurusan = 'Informatika';
17         $mahasiswa->save();
18
19     }
20 }
```

### 3. Input Data menggunakan Eloquent ORM

```
app > Models > Mahasiswa.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class mahasiswas extends Model {
8      protected $fillable = [
9          'nim',
10         'nama_lengkap',
11         'tempat_lahir',
12         'tanggal_lahir',
13         'fakultas',
14         'jurusan',
15         'alamat'
16     ];
17 }
```

```

public function insertData() {
    $mahasiswa = new Mahasiswa;
    $mahasiswa->nim = '19003036';
    $mahasiswa->nama_lengkap = 'Muhammad Hamada';
    $mahasiswa->tempat_lahir = 'Bandung';
    $mahasiswa->tanggal_lahir = '2002-02-02';
    $mahasiswa->alamat = 'Jl. Merdeka No. 10';
    $mahasiswa->fakultas = 'Teknik';
    $mahasiswa->jurusan = 'Informatika';
    $mahasiswa->save();
}

}

```

The screenshot shows the MySQL Workbench interface with the 'mahasiswa' table selected. The table has columns: id, nim, nama, tempat\_lahir, tanggal\_lahir, fakultas, jurusan, created\_at, updated\_at, and ipk. The data is as follows:

	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	id	nim	nama	tempat_lahir	tanggal_lahir	fakultas	jurusan	created_at	updated_at	ipk
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	1	20104064	Muhammad Nur Hamada	Purwokerto	2002-02-02	Teknik	Informatika	NULL	NULL	1.00
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	2	20104064	Muhammad Nur Hamada	Purwokerto	2002-02-02	Teknik	Informatika	NULL	NULL	1.00
<input type="checkbox"/>	<input type="button" value="Edit"/>	<input type="button" value="Copy"/>	<input type="button" value="Delete"/>	3	20104064	Muhammad Nur Hamada	Purwokerto	2002-02-02	Teknik	Informatika	NULL	NULL	1.00

### Penjelasan:

Laravel berfungsi untuk mengelola struktur database dan melakukan manipulasi data secara terorganisir menggunakan fitur bawaan framework Laravel. Pada bagian migration, kode berfungsi untuk membuat, mengubah, dan menghapus tabel database melalui method up() dan down(). Method up() dijalankan saat perintah php artisan migrate dieksekusi, yang bertugas membangun atau memodifikasi struktur tabel sesuai dengan skema yang didefinisikan menggunakan Schema dan Blueprint. Sementara itu, method down() berfungsi sebagai kebalikan dari up(), yaitu mengembalikan perubahan struktur database saat proses rollback dijalankan. Mekanisme ini memungkinkan pengembang melakukan perubahan database secara aman, terkontrol, dan dapat dikembalikan ke kondisi sebelumnya jika terjadi kesalahan.

Pada pengolahan data, Laravel menyediakan tiga pendekatan utama, yaitu DB Facade (Raw SQL), Query Builder, dan Eloquent ORM. DB Facade

memungkinkan kode menjalankan perintah SQL secara langsung ke database, sehingga cocok digunakan untuk query sederhana maupun kompleks. Query Builder bekerja dengan cara menyusun query SQL menggunakan method Laravel, yang membuat kode lebih aman dan mengurangi risiko SQL Injection. Sementara itu, Eloquent ORM bekerja dengan merepresentasikan tabel database sebagai sebuah model berbasis objek, sehingga setiap data diperlakukan sebagai objek yang dapat diakses dan dimanipulasi melalui properti dan method. Ketiga pendekatan ini memberikan fleksibilitas kepada pengembang dalam memilih metode pengolahan data sesuai kebutuhan aplikasi, sekaligus menjaga keamanan, efisiensi, dan keterbacaan kode program.

## B. UNGUIDED (Tugas Mandiri)

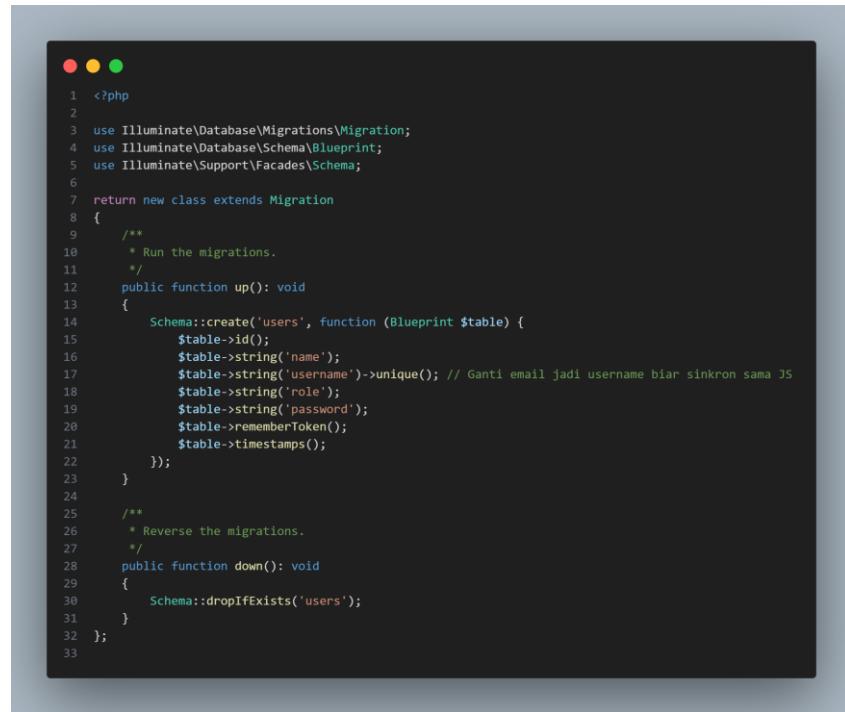
(Bagian ini berisi latihan atau tugas tambahan yang diberikan di akhir modul.

Tuliskan nomor soal, isi soalnya, source code jawaban, screenshot output, dan penjelasan setiap nomor.)

### 1. Soal 1:

Buatlah file migration untuk database yang akan digunakan pada tugas besar di kelas teori.

Class database/migration/2014\_10\_12\_000000\_create\_users\_table.php



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('users', function (Blueprint $table) {
15             $table->id();
16             $table->string('name');
17             $table->string('username')->unique(); // Ganti email jadi username biar sinkron sama JS
18             $table->string('role');
19             $table->string('password');
20             $table->rememberToken();
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      */
28     public function down(): void
29     {
30         Schema::dropIfExists('users');
31     }
32 };
33 
```

## Class migration/2014\_10\_12\_100000\_create\_password\_reset\_tokens\_table.php



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('password_reset_tokens', function (Blueprint $table) {
15             $table->string('email')->primary();
16             $table->string('token');
17             $table->timestamp('created_at')->nullable();
18         });
19     }
20
21     /**
22      * Reverse the migrations.
23      */
24     public function down(): void
25     {
26         Schema::dropIfExists('password_reset_tokens');
27     }
28 };
29
```

## Class database/migration/2019\_08\_19\_000000\_create\_failed\_jobs\_table.php



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('failed_jobs', function (Blueprint $table) {
15             $table->id();
16             $table->string('uuid')->unique();
17             $table->text('connection');
18             $table->text('queue');
19             $table->longText('payload');
20             $table->longText('exception');
21             $table->timestamp('failed_at')->useCurrent();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      */
28     public function down(): void
29     {
30         Schema::dropIfExists('failed_jobs');
31     }
32 };
33
```

## Class migration/2019\_12\_14\_000001\_create\_personal\_access\_tokens\_table.php



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('personal_access_tokens', function (Blueprint $table) {
15             $table->id();
16             $table->morphs('tokenable');
17             $table->string('name');
18             $table->string('token', 64)->unique();
19             $table->text('abilities')->nullable();
20             $table->timestamp('last_used_at')->nullable();
21             $table->timestamp('expires_at')->nullable();
22             $table->timestamps();
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      */
29     public function down(): void
30     {
31         Schema::dropIfExists('personal_access_tokens');
32     }
33 };
34
```

## Class database/migration2025\_12\_23\_034721\_create\_gurus\_table.php



```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('gurus', function (Blueprint $table) {
15             $table->id();
16             // Relasi ke users (PENTING: ini yang nyambungin ke password/usn di tabel users)
17             $table->foreignId('user_id')->constrained('users')->onDelete('cascade');
18
19             // Data Pribadi Guru
20             $table->string('nip')->unique(); // Nomor Induk Pegawai / ID Guru
21             $table->string('nama_lengkap');
22             $table->string('kelas')->nullable(); // Kelas yang diajar (bisa multiple, pisah koma)
23             $table->enum('jenis_kelamin', ['Laki-laki', 'Perempuan']);
24             $table->string('tempat_lahir');
25             $table->date('tanggal_lahir');
26             $table->text('alamat');
27             $table->string('no_hp');
28             $table->string('foto')->nullable();
29             $table->timestamps();
30         });
31     }
32
33     /**
34      * Reverse the migrations.
35      */
36     public function down(): void
37     {
38         Schema::dropIfExists('gurus');
39     }
40};
```

## Class database/migration/2025\_12\_23\_034831\_create\_siswas\_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('siswas', function (Blueprint $table) {
15             $table->id();
16             // Relasi ke tabel users (Menghubungkan profil dengan akun login)
17             $table->foreignId('user_id')->constrained('users')->onDelete('cascade');
18             $table->string('nis')->unique(); // ID Siswa
19             $table->string('nama_lengkap');
20             $table->string('kelas');
21             $table->enum('jenis_kelamin', ['Laki-laki', 'Perempuan']);
22             $table->string('tempat_lahir');
23             $table->date('tanggal_lahir'); // Pake tipe date buat kalender
24             $table->text('alamat');
25             $table->string('no_hp');
26             $table->string('foto')->nullable();
27             $table->timestamps();
28         });
29     }
30
31     /**
32      * Reverse the migrations.
33      */
34     public function down(): void
35     {
36         //
37     }
38 };
39 
```

## Class database/migration/2025\_12\_25\_054310\_create\_presensis\_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('presensis', function (Blueprint $table) {
15             $table->id();
16
17             // Foreign Keys
18             $table->foreignId('siswa_id')->constrained('siswas')->onDelete('cascade');
19
20             // Data Presensi
21             $table->date('tanggal'); // Tanggal presensi
22             $table->enum('status', ['hadir', 'izin', 'sakit', 'alpha'])->default('alpha');
23             $table->time('waktu')->nullable(); // Waktu absen (jam masuk)
24             $table->text('keterangan')->nullable(); // Keterangan tambahan (misal: alasan izin)
25
26             // Metadata
27             $table->string('dicatat_oleh')->nullable(); // Guru yang mencatat
28
29             $table->timestamps();
30
31             // Index untuk query cepat
32             $table->index(['siswa_id', 'tanggal']);
33             $table->index('tanggal');
34             $table->index('status');
35
36             // Unique constraint: 1 siswa hanya bisa 1 presensi per hari
37             $table->unique(['siswa_id', 'tanggal']);
38         });
39     }
40
41     /**
42      * Reverse the migrations.
43      */
44     public function down(): void
45     {
46         Schema::dropIfExists('presensis');
47     }
48 }; 
```

## Class migration/2025\_12\_26\_081424\_create\_perkembangans\_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11     */
12     public function up(): void
13     {
14         Schema::create('perkembangans', function (Blueprint $table) {
15             $table->id();
16
17             // Foreign Keys
18             $table->foreign('siswa_id')->constrained('siswas')->onDelete('cascade');
19
20             // Data Perkembangan
21             $table->date('tanggal'); // Tanggal pencatatan
22
23             // Tilawati/Bacaan
24             $table->string('tilawati')->nullable(); // Jilid 1-6 / Al-Quran
25             $table->string('halaman')->nullable(); // Halaman berapa
26
27             // Kemampuan/Penilaian
28             $table->enum('kemampuan', ['Sangat Baik', 'Baik', 'Cukup', 'Perlu Bimbingan'])->nullable();
29
30             // Hafalan
31             $table->string('hafalan')->nullable(); // Surah yang dihafal
32             $table->string('syat')->nullable(); // Ayat berapa
33
34             // Perilaku & Catatan
35             $table->text('catat_krama')->nullable(); // Catatan perilaku
36             $table->text('catatan')->nullable(); // Catatan tambahan guru
37
38             // Metadata
39             $table->string('dicatat_oleh')->nullable(); // Guru yang mencatat
40
41             $table->timestamps();
42
43             // Index untuk query cepat
44             $table->index(['siswa_id', 'tanggal']);
45             $table->index(['tanggal']);
46
47             // Unique constraint: 1 siswa 1 catatan per hari
48             $table->unique(['siswa_id', 'tanggal']);
49         });
50     }
51
52     /**
53      * Reverse the migrations.
54     */
55     public function down(): void
56     {
57         Schema::dropIfExists('perkembangans');
58     }
59 };
```

## Class database/migration/2025\_12\_30\_024144\_create\_notifikasi\_table.php

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     public function up(): void
10    {
11        Schema::create('notifikasi', function (Blueprint $table) {
12            $table->id();
13
14            // Penerima
15            $table->string('no_hp'); // Nomor whatsapp tujuan
16            $table->string('nama_penerima'); // Nama penerima
17            $table->enum('tipe_penerima', ['orang_tua', 'guru', 'kelas', 'semua'])->default('orang_tua');
18
19            // Relasi (optional - jika ada siswa_id atau guru_id terikuti)
20            $table->foreign('siswa_id')->nullable()->constrained('siswas')->onDelete('cascade');
21            $table->foreign('guru_id')->nullable()->constrained('users')->onDelete('cascade');
22
23            // Konten Pesan
24            $table->enum('tipe_notifikasi', ['presensi', 'perkembangan', 'manual', 'pengumuman'])->default('manual');
25            $table->text('pesan'); // Isi pesan whatsaApp
26
27            // Status Pengiriman
28            $table->enum('status', ['pending', 'berhasil', 'gagal'])->default('pending');
29            $table->text('error_message')->nullable(); // Jika gagal, simpan error
30            $table->timestamp('sent_at')->nullable(); // Waktu terkirim
31
32            // Metadata
33            $table->string('reference_type')->nullable(); // Presensi, Perkembangan, dkk
34            $table->unsignedInteger('reference_id')->nullable(); // ID dari presensi/perkembangan
35            $table->foreignId('sent_by')->nullable()->constrained('users')->onDelete('set null'); // Admin yang kirim
36
37            $table->timestamps();
38
39            // Indexes
40            $table->index('no_hp');
41            $table->index('status');
42            $table->index('tipe_notifikasi');
43            $table->index(['reference_type', 'reference_id']);
44        });
45    }
46
47    public function down(): void
48    {
49        Schema::dropIfExists('notifikasi');
50    }
51 };
```

## Output:

The screenshot shows the phpMyAdmin interface for the db\_toppmart database. The 'Tables' section displays 10 tables: failed\_jobs, gurus, migrations, nomenclature, password\_resets, personal\_access\_tokens, presensi, silex, and users. The 'Create new table' form is open at the bottom, showing a table name 'Console' and 4 columns.

Table users

The screenshot shows the detailed structure of the 'users' table in the db\_toppmart database. The table has 11 columns: id, name, username, role, password, remember\_token, created\_at, and updated\_at. Primary key 'id' and unique index 'username' are defined.

Table gurus

The screenshot shows the detailed structure of the 'gurus' table in the db\_toppmart database. The table has 13 columns: id, user\_id, name, name\_length, kelas, jenis\_kelamin, tempat\_lahir, tanggal\_lahir, alamat, no\_hp, tgl, and updated\_at. Primary key 'id' and unique index 'gurus\_nip\_unique' are defined.

Table siswas

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	<code>bigint(20)</code>		<code>UNSIGNED</code>	No	<code>None</code>			
2	<code>user_id</code>	<code>bigint(20)</code>		<code>UNSIGNED</code>	No	<code>None</code>			
3	<code>nis</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
4	<code>name_jangka</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
5	<code>kelas</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
6	<code>jenjang</code>	<code>enum('SD','MI','SMP','MTS','SMA')</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
7	<code>tempat_lahir</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
8	<code>tanggal_lahir</code>	<code>date</code>			No	<code>None</code>			
9	<code>alamat</code>	<code>text</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
10	<code>no_hp</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
11	<code>foto</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>None</code>			
12	<code>created_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			
13	<code>updated_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			

Indexes:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
	<code>PRIMARY</code>		BTREE	Yes	<code>id</code>	1	A	No	
	<code>siswas_ns_unique</code>	BTREE	Yes	No	<code>nis</code>	1	A	No	
	<code>siswas_user_id_foreign</code>	BTREE	No	No	<code>user_id</code>	1	A	No	

Table notifikasi

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	<code>bigint(20)</code>		<code>UNSIGNED</code>	No	<code>None</code>			
2	<code>hp</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
3	<code>nama_penerima</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
4	<code>tipe_notifikasi</code>	<code>enum('pesan','status','error')</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
5	<code>pesan</code>	<code>text</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
6	<code>status</code>	<code>enum('pending','berhasil','cancel')</code>		<code>utf8mb4_unicode_ci</code>	No	<code>None</code>			
7	<code>error_message</code>	<code>text</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
8	<code>sent_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			
9	<code>reference_id</code>	<code>varchar(255)</code>			Yes	<code>NULL</code>			
10	<code>reference_id</code>	<code>bigint(20)</code>			Yes	<code>NULL</code>			
11	<code>sent_by</code>	<code>bigint(20)</code>			Yes	<code>NULL</code>			
12	<code>created_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			
13	<code>updated_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			

Indexes:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
	<code>PRIMARY</code>		BTREE	Yes	<code>id</code>	1	A	No	
	<code>notifikasi_ns_unique</code>	BTREE	Yes	No	<code>id</code>	1	A	No	
	<code>notifikasi_user_id_foreign</code>	BTREE	No	No	<code>reference_id</code>	1	A	No	

Table perkembangans

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	<code>bigint(20)</code>		<code>UNSIGNED</code>	No	<code>None</code>			
2	<code>siswa_id</code>	<code>bigint(20)</code>		<code>UNSIGNED</code>	No	<code>None</code>			
3	<code>tanggap_id</code>	<code>bigint(20)</code>		<code>UNSIGNED</code>	No	<code>None</code>			
4	<code>diwasi_id</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
5	<code>halaman</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
6	<code>kemampuan</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
7	<code>halaman_kemampuan</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
8	<code>ayat</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
9	<code>tata_krama</code>	<code>text</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
10	<code>catatan</code>	<code>text</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
11	<code>diskusi_diklat</code>	<code>varchar(255)</code>		<code>utf8mb4_unicode_ci</code>	Yes	<code>NULL</code>			
12	<code>created_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			
13	<code>updated_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>			

Indexes:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
	<code>PRIMARY</code>		BTREE	Yes	<code>id</code>	1	A	No	
	<code>perkembangans_siswa_id_tanggap_id_diwasi_id_foreign</code>	BTREE	Yes	No	<code>siswa_id</code> , <code>tanggap_id</code> , <code>diwasi_id</code>	3	A	No	
	<code>perkembangans_kemampuan_id_unique</code>	BTREE	Yes	No	<code>kemampuan</code>	1	A	No	

Table presensi

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	<code>bigint(20)</code>		<code>auto_increment</code>	No	<code>int(11)</code>	AUTO_INCREMENT	<code>VISIBLE, NO_DROP</code>	<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop
2	<code>siswa_id</code>	<code>int(20)</code>			No	<code>int(11)</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
3	<code>tanggal</code>				No	<code>date</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
4	<code>status</code>	<code>enum('absent','izin','sakit','alih')</code>			No	<code>enum('absent','izin','sakit','alih')</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
5	<code>waktu</code>	<code>time</code>			Yes	<code>NULL</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
6	<code>keterangan</code>	<code>text</code>		<code>attribute:enctype:ct</code>	No	<code>text</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
7	<code>dicatat_siswa</code>	<code>varchar(255)</code>		<code>attribute:enctype:ct</code>	Yes	<code>text</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
8	<code>created_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	
9	<code>updated_at</code>	<code>timestamp</code>			Yes	<code>NULL</code>		<input checked="" type="checkbox"/> Change <input type="checkbox"/> Drop	

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
<input checked="" type="checkbox"/> Create	<input checked="" type="checkbox"/> Rename <input type="checkbox"/> Drop	PRIMARY	BTREE	Yes	<code>id</code>	0	A	No	
<input checked="" type="checkbox"/> Create	<input checked="" type="checkbox"/> Rename <input type="checkbox"/> Drop	presensi_siswa_id_tanggal_unique	HASH	Yes	<code>siswa_id</code>	0	A	No	
<input checked="" type="checkbox"/> Create	<input checked="" type="checkbox"/> Rename <input type="checkbox"/> Drop	presensi_siswa_id_tanggal_index	BTREE	No	<code>siswa_id</code>	0	A	No	
<input checked="" type="checkbox"/> Create	<input checked="" type="checkbox"/> Rename <input type="checkbox"/> Drop	presensi_tanggal_index	BTREE	No	<code>tanggal</code>	0	A	No	
<input checked="" type="checkbox"/> Create	<input checked="" type="checkbox"/> Rename <input type="checkbox"/> Drop	presensi_status_index	BTREE	No	<code>status</code>	0	A	No	

### Penjelasan:

Migration tersebut berfungsi sebagai mekanisme untuk membangun, mengubah, dan mengelola struktur basis data aplikasi Laravel secara sistematis. Setiap migration berisi definisi pembuatan tabel tertentu, baik tabel bawaan Laravel seperti users, password\_reset\_tokens, failed\_jobs, dan personal\_access\_tokens, maupun tabel khusus aplikasi seperti guru, siswa, presensi, perkembangan, dan notifikasi. Di dalam setiap file, method up() digunakan untuk mendefinisikan struktur tabel, termasuk nama kolom, tipe data, primary key, serta relasi antar tabel menggunakan foreign key. Ketika perintah php artisan migrate dijalankan, Laravel akan mengeksekusi migration tersebut secara berurutan berdasarkan timestamp pada nama file, sehingga database terbentuk sesuai rancangan sistem. Sebaliknya, method down() memungkinkan penghapusan tabel saat dilakukan rollback, sehingga perubahan database dapat dikontrol dan dikembalikan ke kondisi sebelumnya. Dengan penggunaan migration, pengelolaan struktur database menjadi lebih rapi, konsisten, dan memudahkan kolaborasi dalam pengembangan aplikasi.

## 2. Soal 2:

Buatlah fungsi-fungsi berikut untuk menyelesaikan tugas besar di matakuliah teori:

- 1) Buat fungsi untuk insert data menggunakan raw SQL Queries
- 2) Buat fungsi untuk insert data menggunakan Query Builder
- 3) Buat fungsi untuk insert data menggunakan Eloquent ORM

## Jawaban:

## 1) raw SQL Queries

```
1 // ...  
2 // INSERT DATA (RAW SQL)  
3 public function insertRawSql()  
4 {  
5     $db=>beginTransaction();  
6  
7     try {  
8         // 1. Insert ke tabel users (parent)  
9         $db->insert(  
10             "INSERT INTO users (name,username,password,role,created_at,updated_at)  
11             VALUES (:name,:username,:password,:role,:created_at,:updated_at)",  
12             array('name'=>$name,'username'=>$username,'password'=>$password,'role'=>$role,'created_at'=>$now,'updated_at'=>$now));  
13  
14         // Show Raw SQL  
15         //rawsql_user_ . time());  
16         //die($db->getLastQuery());  
17         //time);  
18     }  
19  
20     // 2. Ambil ID user terakhir  
21     $user_id = $db->lastInsertId();  
22  
23     // 3. Insert ke tabel times (child)  
24     $db->insert(  
25         "INSERT INTO times  
26         (user_id,mis,nama_lengkap,relasi,jenis_kelamin,tempat_lahir,tanggal_lahir,alamat,no_hp,created_at,updated_at)  
27         VALUES (:user_id,:mis,:nama_lengkap,:relasi,:jenis_kelamin,:tempat_lahir,:tanggal_lahir,:alamat,:no_hp,:created_at,:updated_at)",  
28         array('user_id'=>$user_id,'mis'=>$mis,'nama_lengkap'=>$nama_lengkap,'relasi'=>$relasi,'jenis_kelamin'=>$jenis_kelamin,'tempat_lahir'=>$tempat_lahir,'tanggal_lahir'=>$tanggal_lahir,'alamat'=>$alamat,'no_hp'=>$no_hp,'created_at'=>$now,'updated_at'=>$now));  
29  
30         // Show Raw SQL  
31         //rawsql_time_ . time());  
32         //die($db->getLastQuery());  
33         //time);  
34  
35     }  
36  
37     $db->commit();  
38  
39     return response()->json([  
40         'success' => true,  
41         'message' => 'Data berhasil diinsert',  
42         'data' => $user_id  
43     ]);  
44 } catch (\Exception $e) {  
45     $db->rollBack();  
46  
47     return response()->json([  
48         'success' => false,  
49         'error' => $e->getMessage(),  
50         'code' => 500  
51     ]);  
52 }
```

## 2) Query Builder

```
1 // INSERT DATA (QUERY BUILDER)
2 public function insertQueryBuilder()
3 {
4     DB::beginTransaction();
5
6     try {
7         // 1. Insert users
8         $userId = DB::table('users')->insertGetId([
9             'name'      => 'Siswa Query Builder',
10            'username'   => 'qb_user_'. time(),
11            'password'   => Hash::make('password123'),
12            'role'       => 'siswa',
13            'created_at' => now(),
14            'updated_at' => now(),
15        ]);
16
17         // 2. Insert siswas
18         DB::table('siswas')->insert([
19             'user_id'    => $userId,
20             'nis'        => 'Q8'. rand(100, 999),
21             'nama_lengkap' => 'Siswa Query Builder',
22             'kelas'      => '1B',
23             'jenis_kelamin' => 'Perempuan',
24             'tempat_lahir' => 'Binjai',
25             'tanggal_lahir' => '2011-02-02',
26             'alamat'     => 'Alamat Query Builder',
27             'no_hp'       => '081298765432',
28             'created_at'  => now(),
29             'updated_at'  => now(),
30         ]);
31
32         DB::commit();
33
34         return response()->json([
35             'success' => true,
36             'method'  => 'Query Builder',
37             'message' => 'Insert data menggunakan Query Builder berhasil'
38         ]);
39     } catch (\Exception $e) {
40         DB::rollBack();
41
42         return response()->json([
43             'success' => false,
44             'error'   => $e->getMessage()
45         ], 500);
46     }
47 }
```

### 3) Eloquent ORM



```
1 // INSERT DATA (ELOQUENT ORM)
2     public function insertEloquent()
3     {
4         DB::beginTransaction();
5
6         try {
7             $user = User::create([
8                 'name' => 'Siswa Eloquent ORM',
9                 'username' => 'eloquent_user_'. time(),
10                'password' => Hash::make('password123'),
11                'role' => 'siswa',
12            ]);
13
14            Siswa::create([
15                'user_id' => $user->id,
16                'nis' => 'EL'. rand(100, 999),
17                'nama_lengkap' => 'Siswa Eloquent ORM',
18                'kelas' => '1C',
19                'jenis_kelamin' => 'Laki-laki',
20                'tempat_lahir' => 'Medan',
21                'tanggal_lahir' => '2012-03-03',
22                'alamat' => 'Alamat Eloquent',
23                'no_hp' => '081200000000',
24            ]);
25
26            DB::commit();
27
28            return response()->json([
29                'success' => true,
30                'method' => 'Eloquent ORM',
31                'message' => 'Insert data menggunakan Eloquent ORM berhasil'
32            ]);
33        } catch (\Exception $e) {
34            DB::rollBack();
35
36            return response()->json([
37                'success' => false,
38                'error' => $e->getMessage()
39            ], 500);
40        }
41    }
42 }
```

#### Output:

raw SQL Queries



← ⌂ ⓘ 127.0.0.1:8000/insert/siswa/eloquent

Pretty-print □

```
{"success":true,"method":"Eloquent ORM","message":"Insert data menggunakan Eloquent ORM berhasil"}
```

#### QueryBuilder



← ⌂ ⓘ 127.0.0.1:8000/insert/siswa/query-builder

Pretty-print □

```
{"success":true,"method":"Query Builder","message":"Insert data menggunakan Query Builder berhasil"}
```

#### Eloquent ORM



← ⌂ ⓘ 127.0.0.1:8000/insert/siswa/eloquent

Pretty-print □

```
{"success":true,"method":"Eloquent ORM","message":"Insert data menggunakan Eloquent ORM berhasil"}
```

### **Penjelasan:**

Kode yang dibuat berfungsi untuk melakukan proses penyimpanan (insert) data siswa ke dalam database dengan tiga pendekatan yang berbeda, yaitu Raw SQL Queries, Query Builder, dan Eloquent ORM. Ketiga fungsi tersebut berada di dalam SiswaController dan masing-masing menjalankan proses penyimpanan data ke dua tabel yang saling berelasi, yaitu tabel users sebagai parent dan tabel siswas sebagai child yang terhubung melalui user\_id.

Pada metode Raw SQL Queries, proses insert dilakukan dengan menuliskan perintah SQL secara manual menggunakan DB::insert(). Setelah data user disimpan, sistem mengambil id terakhir menggunakan lastInsertId() untuk kemudian digunakan sebagai foreign key saat menyimpan data siswa. Seluruh proses dibungkus dalam transaksi database (DB::beginTransaction) untuk menjaga konsistensi data apabila terjadi kesalahan.

Pada metode Query Builder, Laravel menyediakan cara yang lebih aman dan terstruktur tanpa menuliskan SQL secara langsung. Proses insert dilakukan menggunakan DB::table()->insertGetId() untuk mendapatkan id user, lalu dilanjutkan dengan insert data siswa. Pendekatan ini mengurangi risiko kesalahan sintaks SQL serta lebih mudah dibaca dibandingkan Raw SQL.

Sedangkan pada metode Eloquent ORM, proses insert dilakukan menggunakan model User dan Siswa secara langsung melalui method create(). Laravel secara otomatis menangani relasi antar tabel dan proses penyimpanan data. Pendekatan ini menghasilkan kode yang lebih singkat, rapi, dan mudah dipelihara, sehingga sangat cocok digunakan pada pengembangan aplikasi berskala besar.

## **BAB III**

### **PENUTUP**

#### **A. Kesimpulan**

Berdasarkan hasil praktikum dan tugas mandiri pada Modul 12, dapat disimpulkan bahwa penggunaan migration dan model pada framework Laravel sangat membantu dalam pengelolaan struktur database dan manipulasi data secara terstruktur, aman, dan terkontrol. Praktikum ini memberikan pemahaman mengenai pembuatan dan modifikasi tabel database menggunakan migration, serta penerapan tiga pendekatan pengolahan data yaitu Raw SQL, Query Builder, dan Eloquent ORM. Kendala yang dihadapi terutama berkaitan dengan kesalahan penulisan struktur migration dan relasi antar tabel, namun dapat diatasi dengan memahami fungsi method up() dan down(), serta melakukan rollback dan perbaikan kode. Secara keseluruhan, praktikum ini meningkatkan pemahaman dalam perancangan database dan implementasi pengolahan data yang efektif pada aplikasi Laravel.