



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

JOBSHEET 03

MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM

Sebelumnya kita sudah membahas mengenai *Routing*, *Controller*, dan *View* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Untuk itu, kita memerlukan teknik untuk merancang/membuat table basis data sebelum membuat aplikasi. Laravel memiliki fitur dalam pengelolaan basis data seperti, migration, seeder, model, dll.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS)*, sesuai dengan **Studi Kasus PWL.pdf**.

Jadi kita bikin project Laravel 10 dengan nama **PWL_POS**.

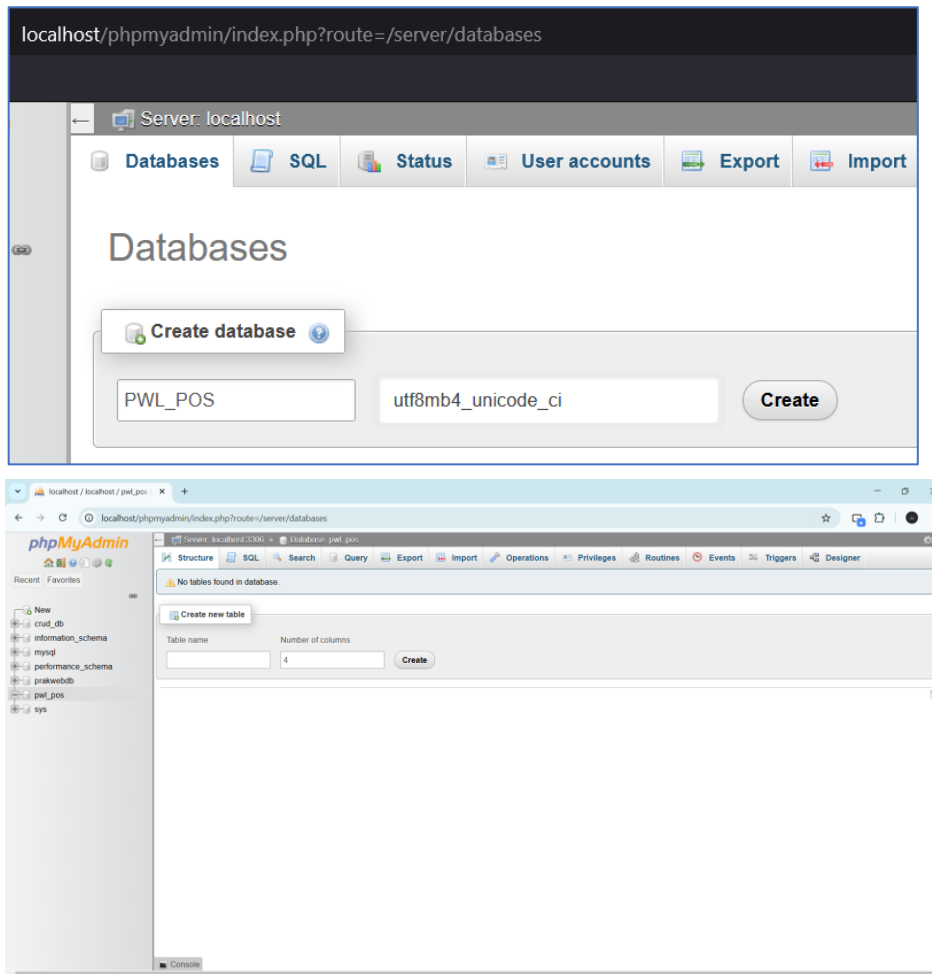
Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. PENGATURAN DATABASE

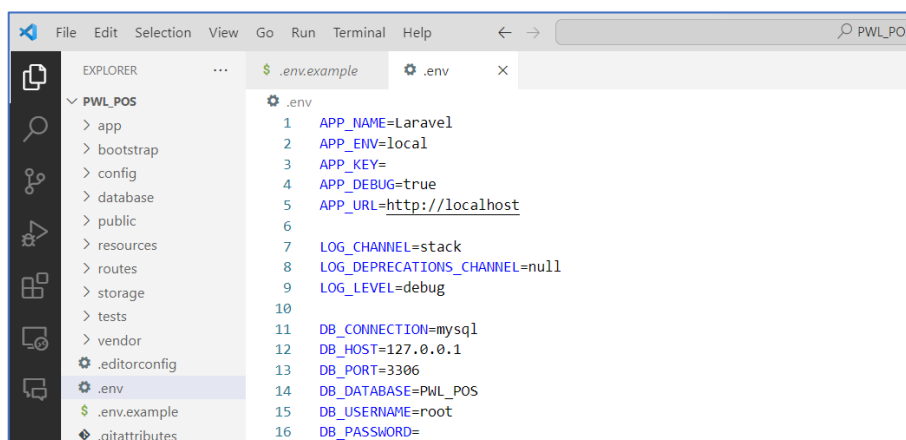
Database atau basis data menjadi komponen penting dalam membangun sistem. Hal ini dikarenakan database menjadi tempat untuk menyimpan data-data transaksi yang ada pada sistem. Koneksi ke database perlu kita atur agar sesuai dengan database yang kita gunakan.

Praktikum 1 - pengaturan database:

1. Buka aplikasi phpMyAdmin, dan buat database baru dengan nama **PWL_POS**



2. Buka aplikasi VSCode dan buka folder project **PWL_POS** yang sudah kita buat
3. Copy file **.env.example** menjadi **.env**
4. Buka file **.env**, dan pastikan konfigurasi **APP_KEY** bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan **php artisan**.



5. Edit file **.env** dan sesuaikan dengan database yang telah dibuat



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2FxJey3lHh13EFasEJDuxXn9Af22Y=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=PWL_POS
15 DB_USERNAME=root
16 DB_PASSWORD=
```

6. Laporkan hasil Praktikum-1 ini dan *commit* perubahan pada *git*.

B. MIGRATION

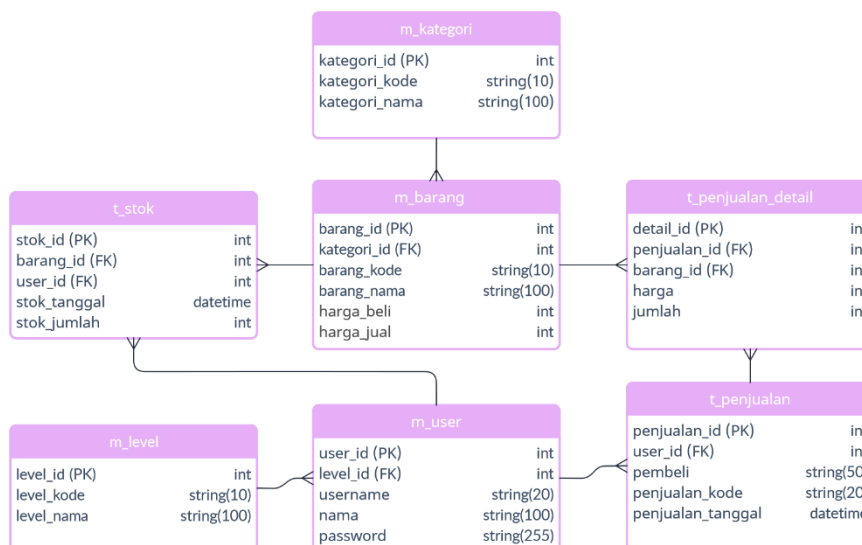
Migration pada Laravel merupakan sebuah fitur yang dapat membantu kita mengelola database secara efisien dengan menggunakan kode program. Migration membantu kita dalam membuat (*create*), mengubah (*edit*), dan menghapus (*delete*) struktur tabel dan kolom pada database yang sudah kita buat dengan cepat dan mudah. Dengan Migration, kita juga dapat melakukan perubahan pada struktur database tanpa harus menghapus data yang ada.

Salah satu keunggulan menggunakan migration adalah mempermudah proses instalasi aplikasi kita, Ketika aplikasi yang kita buat akan diimplementasikan di server/komputer lain.



Sesuai dengan topik pembelajaran kita untuk membangun sistem *Point of Sales (PoS)* sederhana, maka kita perlu membuat migration sesuai desain database yang sudah didefinisikan pada file

Studi Kasus PWL.pdf



Dalam membuat file migration di Laravel, yang perlu kita perhatikan adalah struktur table yang ingin kita buat.

TIPS MIGRATION

Buatlah file migration untuk table yang tidak memiliki relasi (table yang tidak ada *foreign key*) dulu, dan dilanjutkan dengan membuat file migrasi yang memiliki relasi yang sedikit, dan lanjut ke file migrasi dengan table yang memiliki relasi yang banyak.

Dari tips di atas, kita dapat melakukan cek untuk desain database yang sudah ada dengan mengetahui jumlah *foreign key* yang ada. Dan kita bisa menentukan table mana yang akan kita buat migrasinya terlebih dahulu.

No Urut	Nama Tabel	Jumlah FK
1	m_level	0
2	m_kategori	0
3	m_user	1
4	m_barang	1
5	t_penjualan	1
6	t_stok	2
7	t_penjualan_detail	2

INFO



Secara default Laravel sudah ada table **users** untuk menyimpan data pengguna, tapi pada praktikum ini, kita gunakan table sesuai dari file **Studi Kasus PWL.pdf** yaitu **m_user**.

Pembuatan file migrasi bisa menggunakan 2 cara, yaitu

- a. Menggunakan **artisan** untuk membuat *file migration*

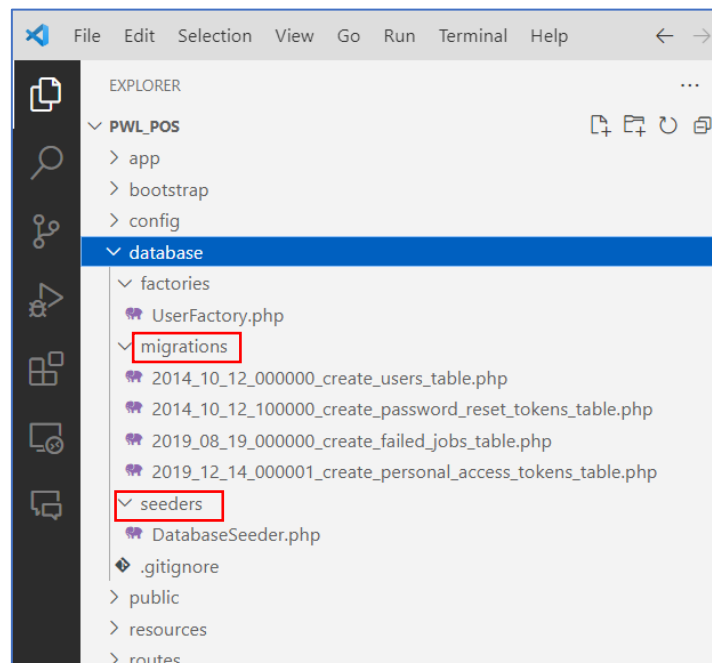
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- b. Menggunakan **artisan** untuk membuat *file model* + *file migration*

```
php artisan make:model <nama-model> -m
```

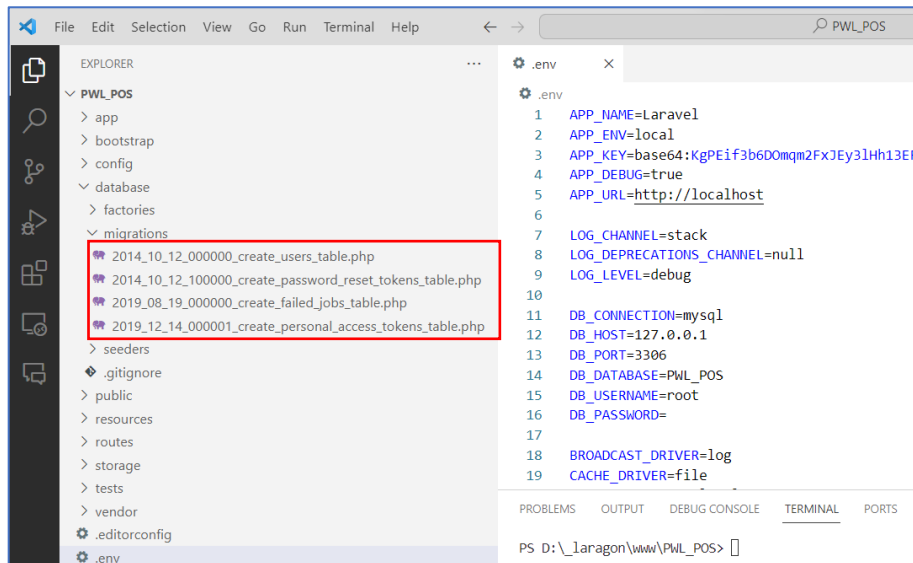
Perintah **-m** di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder **PWL_POS/database**



Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

1. Buka *terminal* VSCode kalian, untuk yang di kotak merah adalah default dari laravel



2. Kita abaikan dulu yang di kotak merah (jangan di hapus)
3. Kita buat file migrasi untuk table `m_level` dengan perintah

```
php artisan make:migration create_m_level_table --create=m_level
```





```
database > migrations > 2025_03_02_163359_create_m_level_table.php > ...
1  k?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('m_level', function (Blueprint $table) {
15               $table->id();
16               $table->timestamps();
17           });
18       }
19
20       /**
21       * Reverse the migrations.
22       */
23       public function down(): void
24       {
25           Schema::dropIfExists('m_level');
26       }
27  };
```

4. Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada

```
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create('m_level', function (Blueprint $table) {
15               $table->id('level_id');
16               $table->string('level_kode', 10)->unique();
17               $table->string('level_nama', 100);
18               $table->timestamps();
19           });
20       }
21
22       /**
23       * Reverse the migrations.
24       */
25       public function down(): void
26       {
27           Schema::dropIfExists('m_level');
28       }
29  };
```

```
public function up(): void
{
    Schema::create('m_level', function (Blueprint $table) {
        $table->id('level_id');
        $table->string('level_kode', 10)->unique();
        $table->string('level_nama', 100);
        $table->timestamps();
    });
}
```



INFO

Dalam fitur migration Laravel, terdapat berbagai macam function untuk membuat kolom di table database. Silahkan cek disini

<https://laravel.com/docs/10.x/migrations#available-column-types>

5. Simpan kode pada tahapan 4 tersebut, kemudian jalankan perintah ini pada terminal VSCode untuk melakukan migrasi

```
php artisan migrate
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\laragon\www\PWL_POS> php artisan migrate

[INFO] Preparing database.

Creating migration table ..... 12ms DONE

[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 16ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 6ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 42ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 15ms DONE
2024_02_25_133526_create_m_level_table ..... 13ms DONE

PS D:\laragon\www\PWL_POS>
```

6. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum

Table	Action	Rows
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	5
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
6 tables	Sum	5	InnoDB	utf8mb4_0900_ai_ci	96.0 KiB	0 B

7. Ok, table sudah dibuat di database
8. Buat table *database* dengan *migration* untuk table **m_kategori** yang sama-sama tidak memiliki *foreign key*



```
database > migrations > 2025_03_02_164231_create_m_kategori_table.php > class > up > Closure
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('m_kategori', function (Blueprint $table) {
15             $table->id('kategori_id', 20);
16             $table->string('kategori_kode', 10)->unique();
17             $table->string('kategori_nama', 100);
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists('m_kategori');
28     }
29 }
30
31
```

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs		0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations		6	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_kategori		0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_level		0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> password_reset_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> personal_access_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> users		0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-

9. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*.

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

1. Buka *terminal* VSCode kalian, dan buat file migrasi untuk table **m_user**

```
php artisan make:migration create_m_user_table --table=m_user
```

2. Buka file migrasi untuk table **m_user**, dan modifikasi seperti berikut



```
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('m_user', function (Blueprint $table) {
15             $table->id('user_id');
16             $table->unsignedBigInteger('level_id')->index(); // indexing untuk ForeignKey
17             $table->string('username', 20)->unique(); // unique untuk memastikan tidak ada username yang sama
18             $table->string('nama', 100);
19             $table->string('password');
20             $table->timestamps();
21
22             // Mendefinisikan Foreign Key pada kolom level_id mengacu pada kolom level_id di tabel m_level
23             $table->foreign('level_id')->references('level_id')->on('m_level');
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('m_user');
33     }
34 };
```

```
database > migrations > 2025_03_02_165548_create_m_user_table.php > class > up > Closure
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('m_user', function (Blueprint $table) {
15             $table->id('user_id');
16             $table->unsignedBigInteger('level_id')->index();
17             $table->string('username', 20)->unique();
18             $table->string('nama', 100);
19             $table->string('password');
20             $table->timestamps();
21
22             $table->foreign('level_id')->references('level_id')->on('m_level');
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      */
29     public function down(): void
30     {
31         Schema::dropIfExists('m_user');
32     }
33 };
```

3. Simpan kode program Langkah 2, dan jalankan perintah **php artisan migrate**. Amati apa yang terjadi pada database.



4. Buat table *database* dengan *migration* untuk table-table yang memiliki *foreign key*

m_barang
t_penjualan
t_stok
t_penjualan_detail

```
PS C:\laragon\www\PM_2025\Winggu3\Jobsheet3> php artisan make:migration create_m_barang_table --create=m_barang
INFO Migration [C:\laragon\www\PM_2025\Winggu3\Jobsheet3\database\migrations\2025_03_02_171022_create_m_barang_table.php] created successfully.
PS C:\laragon\www\PM_2025\Winggu3\Jobsheet3> php artisan make:migration create_t_penjualan_table --create=t_penjualan
INFO Migration [C:\laragon\www\PM_2025\Winggu3\Jobsheet3\database\migrations\2025_03_02_171043_create_t_penjualan_table.php] created successfully.
PS C:\laragon\www\PM_2025\Winggu3\Jobsheet3> php artisan make:migration create_t_stok_table --create=t_stok
INFO Migration [C:\laragon\www\PM_2025\Winggu3\Jobsheet3\database\migrations\2025_03_02_171058_create_t_stok_table.php] created successfully.
PS C:\laragon\www\PM_2025\Winggu3\Jobsheet3> php artisan make:migration create_t_penjualan_detail_table --create=t_penjualan_detail
INFO Migration [C:\laragon\www\PM_2025\Winggu3\Jobsheet3\database\migrations\2025_03_02_171119_create_t_penjualan_detail_table.php] created successfully.
```

```
public function up(): void
{
    Schema::create('m_barang', function (Blueprint $table) {
        $table->id('barang_id');
        $table->unsignedBigInteger('kategori_id');
        $table->string('barang_kode', 10)->unique();
        $table->string('barang_nama', 100);
        $table->integer('harga_beli');
        $table->integer('harga_jual');
        $table->timestamps();

        $table->foreign('kategori_id')->references('kategori_id')->on('m_kategori');
    });
}
```

```
public function up(): void
{
    Schema::create('t_penjualan', function (Blueprint $table) {
        $table->id('penjualan_id');
        $table->unsignedBigInteger('user_id');
        $table->string('pembeli', 50);
        $table->string('penjualan_kode', 20)->unique();
        $table->dateTime('penjualan_tanggal');
        $table->timestamps();

        $table->foreign('user_id')->references('user_id')->on('m_user');
    });
}
```

```
public function up(): void
{
    Schema::create('t_stok', function (Blueprint $table) {
        $table->id('stok_id');
        $table->unsignedBigInteger('barang_id');
        $table->unsignedBigInteger('user_id');
        $table->dateTime('stok_tanggal');
        $table->integer('stok_jumlah');
        $table->timestamps();

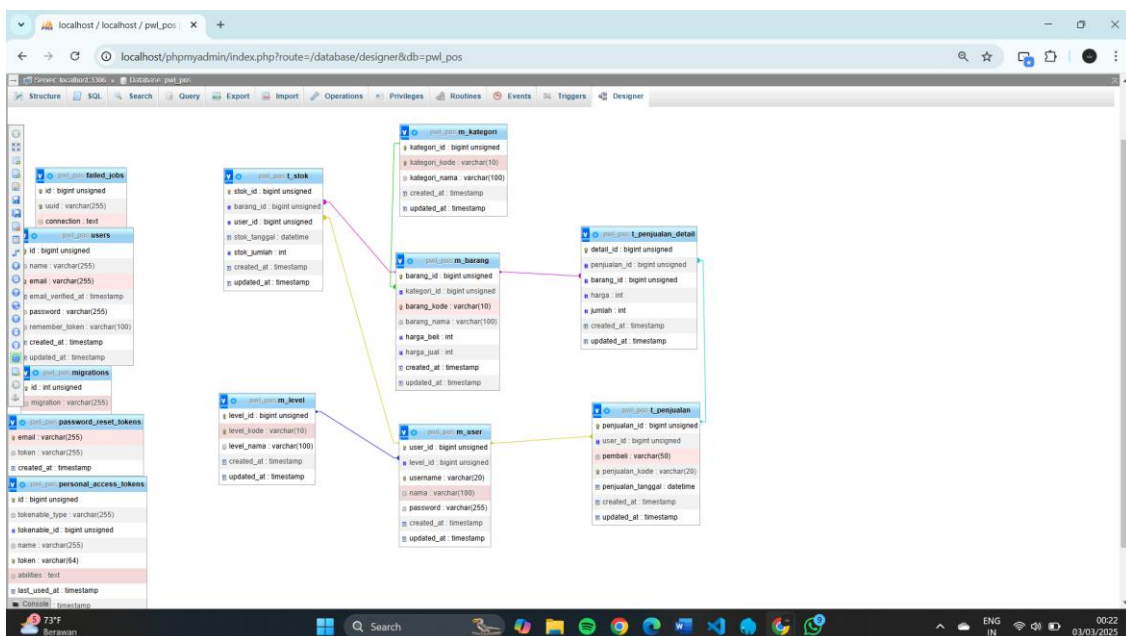
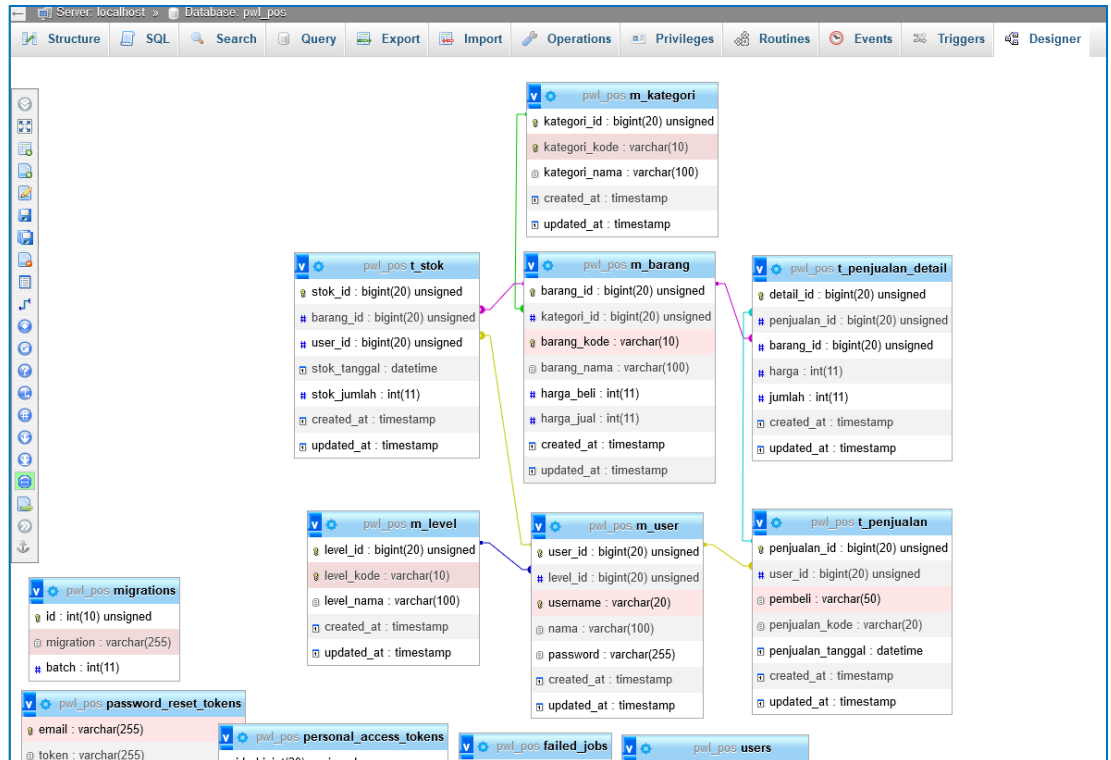
        $table->foreign('barang_id')->references('barang_id')->on('m_barang');
        $table->foreign('user_id')->references('user_id')->on('m_user');
    });
}
```

```
public function up(): void
{
    Schema::create('t_penjualan_detail', function (Blueprint $table) {
        $table->id('detail_id');
        $table->unsignedBigInteger('penjualan_id');
        $table->unsignedBigInteger('barang_id');
        $table->integer('harga');
        $table->integer('jumlah');
        $table->timestamps();

        $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan');
        $table->foreign('barang_id')->references('barang_id')->on('m_barang');
    });
}
```



5. Jika semua file migrasi sudah di buat dan dijalankan maka bisa kita lihat tampilan *designer* pada **phpMyAdmin** seperti berikut



6. Laporkan hasil Praktikum-2.2 ini dan *commit* perubahan pada *git*.



C. SEEDER

Seeder merupakan sebuah fitur yang memungkinkan kita untuk mengisi database kita dengan data awal atau data *dummy* yang telah ditentukan. Seeder memungkinkan kita untuk membuat data awal yang sama untuk setiap penggunaan dalam pembangunan aplikasi. Umumnya, data yang sering dibuat *seeder* adalah data pengguna karena data tersebut akan digunakan saat aplikasi pertama kali di jalankan dan membutuhkan aksi *login*.

1. Perintah umum dalam **membuat file seeder** adalah seperti berikut

```
php artisan make:seeder <nama-class-seeder>
```

Perintah tersebut akan men-generate file seeder pada folder **PWL_POS/database/seeds**

2. Dan perintah untuk **menjalankan file seeder** seperti berikut

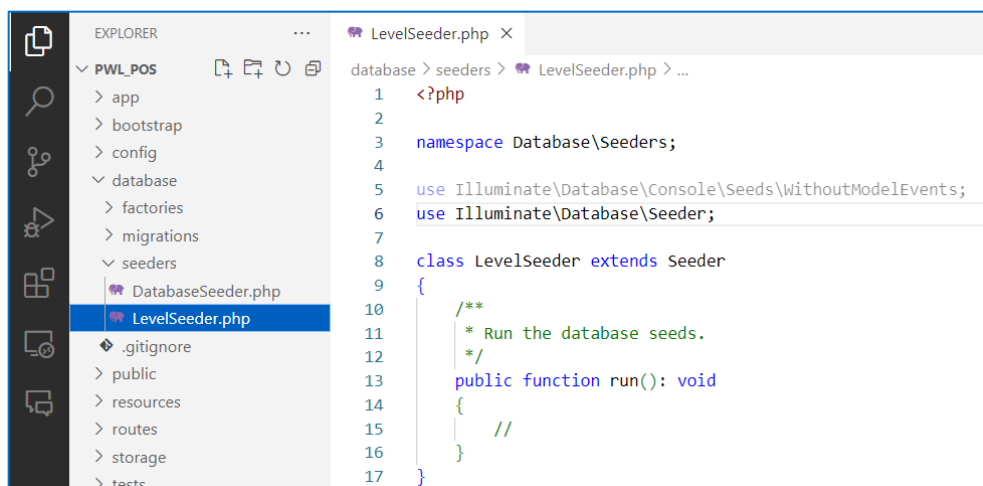
```
php artisan db:seed --class=<nama-class-seeder>
```

Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau *dummy* data untuk memudahkan pengujian dan pengembangan aplikasi kita. Sehingga fitur *seeder* bisa kita pakai dalam membuat sebuah aplikasi web.

Praktikum 3 – Membuat file *seeder*

1. Kita akan membuat file seeder untuk table **m_level** dengan mengetikkan perintah

```
php artisan make:seeder LevelSeeder
```





2. Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function `run()`

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```

3. Selanjutnya, kita jalankan file *seeder* untuk table `m_level` pada terminal

```
php artisan db:seed --class=LevelSeeder
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\_laragon\www\PWL_POS> php artisan db:seed --class=LevelSeeder
INFO Seeding database.
PS D:\_laragon\www\PWL_POS>
```

4. Ketika *seeder* berhasil dijalankan maka akan tampil data pada table `m_level`



				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	STF	Staff/Kasir	NULL	NULL
↑ <input type="checkbox"/> Check all With selected: Edit Copy Delete Export								

				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	STF	Staff/Kasir	NULL	NULL

5. Sekarang kita buat file *seeder* untuk table `m_user` yang me-refer ke table `m_level`

```
php artisan make:seeder UserSeeder
```

6. Modifikasi file `class UserSeeder` seperti berikut

```
9 class UserSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data = [
14             [
15                 'user_id' => 1,
16                 'level_id' => 1,
17                 'username' => 'admin',
18                 'nama' => 'Administrator',
19                 'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
20             ],
21             [
22                 'user_id' => 2,
23                 'level_id' => 2,
24                 'username' => 'manager',
25                 'nama' => 'Manager',
26                 'password' => Hash::make('12345'),
27             ],
28             [
29                 'user_id' => 3,
30                 'level_id' => 3,
31                 'username' => 'staff',
32                 'nama' => 'Staff/Kasir',
33                 'password' => Hash::make('12345'),
34             ],
35         ];
36         DB::table('m_user')->insert($data);
37     }
38 }
```



```
database > seeders > UserSeeder.php > UserSeeder
11 class UserSeeder extends Seeder
12 {
13     /**
14      * Run the database seeds.
15      */
16     public function run(): void
17     {
18         $data = [
19             [
20                 'user_id' => 1,
21                 'level_id' => 1,
22                 'username' => 'admin',
23                 'nama' => 'Administrator',
24                 'password' => Hash::make('12345'),
25             ],
26             [
27                 'user_id' => 2,
28                 'level_id' => 2,
29                 'username' => 'manager',
30                 'nama' => 'Manager',
31                 'password' => Hash::make('12345'),
32             ],
33             [
34                 'user_id' => 3,
35                 'level_id' => 3,
36                 'username' => 'staff',
37                 'nama' => 'Staff/Kasir',
38                 'password' => Hash::make('12345'),
39             ],
40         ];
41         DB::table('m_user')->insert($data);
42     }
43 }
```

7. Jalankan perintah untuk mengeksekusi class `UserSeeder`

```
php artisan db:seed --class=UserSeeder
```

8. Perhatikan hasil seeder pada table `m_user`

	user_id	level_id	username	nama	password
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6fXV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhlFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...
<input type="checkbox"/> Check all With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export					

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$6IMxCVbtJaEoU.dBz8bz6OeN2hMMZ3N0.ngV/a8IKsI...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$5i4qr0q87Zpzo97So.BPeFG8PtUlyxCSLDzCjA1EXE...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$AgRZBp9AganZ9kvDxsCn0.tprRedA41hrdAGrqIM.li...	NULL	NULL

9. Ok, data seeder berhasil di masukkan ke database.
10. Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut

No	Nama Tabel	Jumlah Data	Keterangan
1	<code>m_kategori</code>	5	5 kategori barang
2	<code>m_barang</code>	10	10 barang yang berbeda
3	<code>t_stok</code>	10	Stok untuk 10 barang
4	<code>t_penjualan</code>	10	10 transaksi penjualan
5	<code>t_penjualan_detail</code>	30	3 barang untuk setiap transaksi penjualan



```
PS C:\laragon\www\pwl_2023\Minggu3\Jobsheet3> php artisan make:seeder KategoriSeeder

INFO Seeder [C:\laragon\www\pwl_2023\Minggu3\Jobsheet3\database\seeders\KategoriSeeder.php] created successfully.

PS C:\laragon\www\pwl_2023\Minggu3\Jobsheet3> php artisan make:seeder BarangSeeder

INFO Seeder [C:\laragon\www\pwl_2023\Minggu3\Jobsheet3\database\seeders\BarangSeeder.php] created successfully.

PS C:\laragon\www\pwl_2023\Minggu3\Jobsheet3> php artisan make:seeder StokSeeder

INFO Seeder [C:\laragon\www\pwl_2023\Minggu3\Jobsheet3\database\seeders\StokSeeder.php] created successfully.

PS C:\laragon\www\pwl_2023\Minggu3\Jobsheet3> php artisan make:seeder PenjualanSeeder

INFO Seeder [C:\laragon\www\pwl_2023\Minggu3\Jobsheet3\database\seeders\PenjualanSeeder.php] created successfully.

PS C:\laragon\www\pwl_2023\Minggu3\Jobsheet3> php artisan make:seeder PenjualanDetailSeeder

INFO Seeder [C:\laragon\www\pwl_2023\Minggu3\Jobsheet3\database\seeders\PenjualanDetailSeeder.php] created successfully.
```

```
database > seeders > KategoriSeeder.php > KategoriSeeder > run

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class KategoriSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['kategori_id' => '1', 'kategori_kode' => 'KT001', 'kategori_nama' => "Makanan"],
18             ['kategori_id' => '2', 'kategori_kode' => 'KT002', 'kategori_nama' => "Pakaian"],
19             ['kategori_id' => '3', 'kategori_kode' => 'KT003', 'kategori_nama' => "Snack"],
20             ['kategori_id' => '4', 'kategori_kode' => 'KT004', 'kategori_nama' => "Buah-buahan"],
21             ['kategori_id' => '5', 'kategori_kode' => 'KT005', 'kategori_nama' => "Sayuran"],
22         ];
23
24
25
26 DB::table('m_kategori')->insert($data);
27
28 }
```

```
database > seeders > BarangSeeder.php > BarangSeeder > run

1 namespace Database\Seeders;
2
3 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
4 use Illuminate\Database\Seeder;
5 use Illuminate\Support\Facades\DB;
6
7 class BarangSeeder extends Seeder
8 {
9     /**
10      * Run the database seeds.
11      */
12     public function run(): void
13     {
14         $data = [
15             ['barang_id' => 2, 'kategori_id' => 1, 'barang_kode' => 'BRG002', 'barang_nama' => 'Mie Goreng', 'harga_beli' => 9000, 'harga_jual' => 14000], //
16             ['barang_id' => 3, 'kategori_id' => 1, 'barang_kode' => 'BRG003', 'barang_nama' => 'Ayam Bakar', 'harga_beli' => 12000, 'harga_jual' => 18000], //
17             ['barang_id' => 4, 'kategori_id' => 2, 'barang_kode' => 'BRG004', 'barang_nama' => 'Kaos Polos', 'harga_beli' => 30000, 'harga_jual' => 45000], //
18             ['barang_id' => 5, 'kategori_id' => 2, 'barang_kode' => 'BRG005', 'barang_nama' => 'Celana Jeans', 'harga_beli' => 70000, 'harga_jual' => 100000], //
19             ['barang_id' => 6, 'kategori_id' => 3, 'barang_kode' => 'BRG006', 'barang_nama' => 'Keripik Kentang', 'harga_beli' => 8000, 'harga_jual' => 12000], //
20             ['barang_id' => 7, 'kategori_id' => 3, 'barang_kode' => 'BRG007', 'barang_nama' => 'Biskuit Coklat', 'harga_beli' => 5000, 'harga_jual' => 9000], //
21             ['barang_id' => 8, 'kategori_id' => 4, 'barang_kode' => 'BRG008', 'barang_nama' => 'Apel Fuji', 'harga_beli' => 15000, 'harga_jual' => 20000], //
22             ['barang_id' => 9, 'kategori_id' => 4, 'barang_kode' => 'BRG009', 'barang_nama' => 'Jeruk Manis', 'harga_beli' => 10000, 'harga_jual' => 15000], //
23             ['barang_id' => 10, 'kategori_id' => 5, 'barang_kode' => 'BRG010', 'barang_nama' => 'Wortel Segar', 'harga_beli' => 7000, 'harga_jual' => 10000], //
24             ['barang_id' => 11, 'kategori_id' => 5, 'barang_kode' => 'BRG011', 'barang_nama' => 'Bayam Hijau', 'harga_beli' => 5000, 'harga_jual' => 8000], //
25         ];
26
27
28 DB::table('m_barang')->insert($data);
29
30 }
31
32 }
```



```
database > seeders > StokSeeder.php > ...
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class StokSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data =
14         [
15             ['stok_id' => 1, 'barang_id' => 2, 'user_id' => 1, 'stok_tanggal' => now(), 'stok_jumlah' => 30],
16             ['stok_id' => 2, 'barang_id' => 3, 'user_id' => 1, 'stok_tanggal' => now(), 'stok_jumlah' => 25],
17             ['stok_id' => 3, 'barang_id' => 4, 'user_id' => 2, 'stok_tanggal' => now(), 'stok_jumlah' => 20],
18             ['stok_id' => 4, 'barang_id' => 5, 'user_id' => 2, 'stok_tanggal' => now(), 'stok_jumlah' => 15],
19             ['stok_id' => 5, 'barang_id' => 6, 'user_id' => 2, 'stok_tanggal' => now(), 'stok_jumlah' => 10],
20             ['stok_id' => 6, 'barang_id' => 7, 'user_id' => 3, 'stok_tanggal' => now(), 'stok_jumlah' => 50],
21             ['stok_id' => 7, 'barang_id' => 8, 'user_id' => 3, 'stok_tanggal' => now(), 'stok_jumlah' => 40],
22             ['stok_id' => 8, 'barang_id' => 9, 'user_id' => 3, 'stok_tanggal' => now(), 'stok_jumlah' => 35],
23             ['stok_id' => 9, 'barang_id' => 10, 'user_id' => 1, 'stok_tanggal' => now(), 'stok_jumlah' => 45],
24             ['stok_id' => 10, 'barang_id' => 11, 'user_id' => 1, 'stok_tanggal' => now(), 'stok_jumlah' => 60],
25         ];
26     }
27
28     DB::table('t_stok')->insert($data);
29 }
30 }
```

```
database > seeders > PenjualanSeeder.php > PenjualanSeeder > run
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class PenjualanSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data =
14         [
15             ['penjualan_id' => 1, 'user_id' => 1, 'pembeli' => 'Budi', 'penjualan_kode' => 'P3001', 'penjualan_tanggal' => now()],
16             ['penjualan_id' => 2, 'user_id' => 2, 'pembeli' => 'Siti', 'penjualan_kode' => 'P3002', 'penjualan_tanggal' => now()],
17             ['penjualan_id' => 3, 'user_id' => 3, 'pembeli' => 'Agus', 'penjualan_kode' => 'P3003', 'penjualan_tanggal' => now()],
18             ['penjualan_id' => 4, 'user_id' => 1, 'pembeli' => 'Dewi', 'penjualan_kode' => 'P3004', 'penjualan_tanggal' => now()],
19             ['penjualan_id' => 5, 'user_id' => 2, 'pembeli' => 'Rono', 'penjualan_kode' => 'P3005', 'penjualan_tanggal' => now()],
20             ['penjualan_id' => 6, 'user_id' => 3, 'pembeli' => 'Rina', 'penjualan_kode' => 'P3006', 'penjualan_tanggal' => now()],
21             ['penjualan_id' => 7, 'user_id' => 1, 'pembeli' => 'Slamet', 'penjualan_kode' => 'P3007', 'penjualan_tanggal' => now()],
22             ['penjualan_id' => 8, 'user_id' => 2, 'pembeli' => 'Ani', 'penjualan_kode' => 'P3008', 'penjualan_tanggal' => now()],
23             ['penjualan_id' => 9, 'user_id' => 3, 'pembeli' => 'Eka', 'penjualan_kode' => 'P3009', 'penjualan_tanggal' => now()],
24             ['penjualan_id' => 10, 'user_id' => 1, 'pembeli' => 'Andi', 'penjualan_kode' => 'P3010', 'penjualan_tanggal' => now()],
25         ];
26     }
27
28     DB::table('t_penjualan')->insert($data);
29 }
30 }
```

```
database > seeders > PenjualanDetailSeeder.php > PenjualanDetailSeeder > run
9 class PenjualanDetailSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data =
14         [
15             ['penjualan_id' => 1, 'barang_id' => 2, 'harga' => 14000, 'jumlah' => 2],
16             ['penjualan_id' => 1, 'barang_id' => 3, 'harga' => 18000, 'jumlah' => 1],
17             ['penjualan_id' => 1, 'barang_id' => 4, 'harga' => 45000, 'jumlah' => 3],
18             ['penjualan_id' => 2, 'barang_id' => 5, 'harga' => 100000, 'jumlah' => 1],
19             ['penjualan_id' => 2, 'barang_id' => 6, 'harga' => 12000, 'jumlah' => 4],
20             ['penjualan_id' => 2, 'barang_id' => 7, 'harga' => 9000, 'jumlah' => 2],
21             ['penjualan_id' => 3, 'barang_id' => 8, 'harga' => 20000, 'jumlah' => 2],
22             ['penjualan_id' => 3, 'barang_id' => 9, 'harga' => 15000, 'jumlah' => 3],
23             ['penjualan_id' => 3, 'barang_id' => 10, 'harga' => 10000, 'jumlah' => 2],
24             ['penjualan_id' => 4, 'barang_id' => 11, 'harga' => 8000, 'jumlah' => 5],
25             ['penjualan_id' => 4, 'barang_id' => 2, 'harga' => 14000, 'jumlah' => 1],
26             ['penjualan_id' => 4, 'barang_id' => 3, 'harga' => 18000, 'jumlah' => 2],
27             ['penjualan_id' => 5, 'barang_id' => 4, 'harga' => 45000, 'jumlah' => 1],
28             ['penjualan_id' => 5, 'barang_id' => 5, 'harga' => 100000, 'jumlah' => 3],
29             ['penjualan_id' => 5, 'barang_id' => 6, 'harga' => 12000, 'jumlah' => 2],
30             ['penjualan_id' => 6, 'barang_id' => 7, 'harga' => 9000, 'jumlah' => 4],
31             ['penjualan_id' => 6, 'barang_id' => 8, 'harga' => 20000, 'jumlah' => 2],
32             ['penjualan_id' => 6, 'barang_id' => 9, 'harga' => 15000, 'jumlah' => 1],
33             ['penjualan_id' => 7, 'barang_id' => 10, 'harga' => 10000, 'jumlah' => 3],
34             ['penjualan_id' => 7, 'barang_id' => 11, 'harga' => 8000, 'jumlah' => 2],
35             ['penjualan_id' => 7, 'barang_id' => 2, 'harga' => 14000, 'jumlah' => 4],
36             ['penjualan_id' => 8, 'barang_id' => 3, 'harga' => 18000, 'jumlah' => 1],
37             ['penjualan_id' => 8, 'barang_id' => 4, 'harga' => 45000, 'jumlah' => 2],
38             ['penjualan_id' => 8, 'barang_id' => 5, 'harga' => 100000, 'jumlah' => 1],
39             ['penjualan_id' => 9, 'barang_id' => 6, 'harga' => 12000, 'jumlah' => 2],
40             ['penjualan_id' => 9, 'barang_id' => 7, 'harga' => 9000, 'jumlah' => 3],
41             ['penjualan_id' => 9, 'barang_id' => 8, 'harga' => 20000, 'jumlah' => 2],
42             ['penjualan_id' => 10, 'barang_id' => 9, 'harga' => 15000, 'jumlah' => 4],
43             ['penjualan_id' => 10, 'barang_id' => 10, 'harga' => 10000, 'jumlah' => 1],
44             ['penjualan_id' => 10, 'barang_id' => 11, 'harga' => 8000, 'jumlah' => 2],
45         ];
46     }
47 }
```



Kategori

←T→		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KT003	Snack	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KT004	Buah-buahan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KT005	Sayuran	NULL	NULL

Barang

←T→		barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	2	1	BRG002	Mie Goreng	9000	14000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	1	BRG003	Ayam Bakar	12000	18000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	2	BRG004	Kaos Polos	30000	45000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	2	BRG005	Celana Jeans	70000	100000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	3	BRG006	Keripik Kentang	8000	12000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	3	BRG007	Biskuit Coklat	5000	9000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	4	BRG008	Apel Fuji	15000	20000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	4	BRG009	Jeruk Manis	10000	15000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	5	BRG010	Wortel Segar	7000	10000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	11	5	BRG011	Bayam Hijau	5000	8000	NULL	NULL

Stok

←T→		stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	2	1	2025-03-02 17:48:40	30	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	3	1	2025-03-02 17:48:40	25	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	4	2	2025-03-02 17:48:40	20	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	5	2	2025-03-02 17:48:40	15	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	6	2	2025-03-02 17:48:40	10	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	7	3	2025-03-02 17:48:40	50	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	8	3	2025-03-02 17:48:40	40	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	9	3	2025-03-02 17:48:40	35	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	10	1	2025-03-02 17:48:40	45	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	11	1	2025-03-02 17:48:40	60	NULL	NULL

Penjualan

←T→		penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	Budi	PJ001	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	2	Siti	PJ002	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	3	Agus	PJ003	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	1	Dewi	PJ004	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	2	Tono	PJ005	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	3	Rina	PJ006	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	1	Slamet	PJ007	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	2	Ani	PJ008	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	3	Eka	PJ009	2025-03-02 17:48:47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	1	Andi	PJ010	2025-03-02 17:48:47	NULL	NULL



Penjualan Detail

		detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	2	14000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	1	3	18000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	1	4	45000	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	2	5	100000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	2	6	12000	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	2	7	9000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	3	8	20000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	3	9	15000	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	3	10	10000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	4	11	8000	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	11	4	2	14000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	12	4	3	18000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	13	5	4	45000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	14	5	5	100000	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	15	5	6	12000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	16	6	7	9000	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	17	6	8	20000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	18	6	9	15000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	19	7	10	10000	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	20	7	11	8000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	21	7	2	14000	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	22	8	3	18000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	23	8	4	45000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	24	8	5	100000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	25	9	6	12000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	26	9	7	9000	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	27	9	8	20000	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	28	10	9	15000	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	29	10	10	10000	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	30	10	11	8000	2	NULL	NULL

11. Jika sudah, laporkan hasil Praktikum-3 ini dan *commit* perubahan pada *git*

D. DB FACADE

DB Façade merupakan fitur dari Laravel yang digunakan untuk melakukan *query* secara langsung dengan mengetikkan perintah SQL secara utuh (*raw query*). Disebut *raw query* (query mentah) karena penulisan query pada DB Façade langsung ditulis sebagaimana yang biasa dituliskan pada database, seperti “`select * from m_user`” atau “`insert into m_user...`” atau “`update m_user set ... Where ...`”

Raw query adalah cara paling dasar dan tradisional yang ada di Laravel. Raw query terasa familiar karena biasa kita pakai ketika melakukan query langsung ke database.

INFO

Dokumentasi penggunaan DB Façade bisa dicek di laman ini

<https://laravel.com/docs/10.x/database#running-queries>

Terdapat banyak method yang bisa digunakan pada DB Façade ini. Akan tetapi yang kita pelajari cukup 4 (empat) method yang umum dipakai, yaitu



a. `DB::select()`

Method ini digunakan untuk mengambil data dari database. Method ini **mengembalikan (*return*)** data hasil *query*. Contoh

```
DB::select('select * from m_user'); //Query semua data pada tabel m_user
```

```
DB::select('select * from m_user where level_id = ?', [1]); //Query tabel m_user dengan level_id = 1
```

```
DB::select('select * from m_user where level_id = ? and username = ?', [1, 'admin']);
```

b. `DB::insert()`

Method ini digunakan untuk memasukkan data pada table database. Method ini **tidak memiliki nilai pengembalian (*no return*)**. Contoh

```
DB::insert('insert into m_level(level_kode, level_nama) values(?,?)', ['CUS', 'Pelanggan']);
```

c. `DB::update()`

Method ini digunakan saat menjalankan *raw query* untuk meng-update data pada database. Method ini **memiliki nilai pengembalian (*return*)** berupa jumlah baris data yang ter-update. Contoh

```
DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
```

d. `DB::delete()`

Method ini digunakan saat menjalankan *raw query* untuk menghapus data dari table. Method ini **memiliki nilai pengembalian (*return*)** berupa jumlah baris data yang telah dihapus. Contoh

```
DB::delete('delete from m_level where level_kode = ?', ['CUS']);
```

Ok, sekarang mari kita coba praktikkan menggunakan DB Façade pada project kita

Praktikum 4 – Implementasi DB Facade

1. Kita buat controller dahulu untuk mengelola data pada table `m_level`

```
php artisan make:controller LevelController
```

2. Kita modifikasi dulu untuk *routing*-nya, ada di `PWL_POS/routes/web.php`








```
LevelController.php  web.php X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\LevelController;
4  use Illuminate\Support\Facades\Route;
5
6
7  Route::get('/', function () {
8      return view('welcome');
9  });
10
11 Route::get('/level', [LevelController::class, 'index']);
```

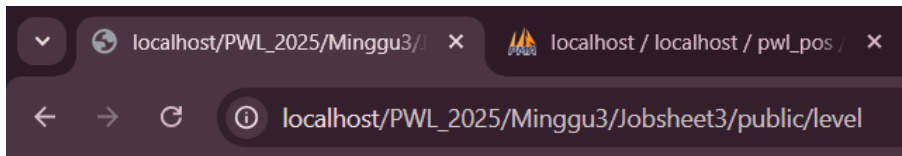
3. Selanjutnya, kita modifikasi file `LevelController` untuk menambahkan 1 data ke table `m_level`

```
LevelController.php X  web.php
app > Http > Controllers > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12         DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13
14         return 'Insert data baru berhasil';
15     }
16 }
```

```
app > Http > Controllers > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index(){
11         DB::insert('insert into a m_level(level_kode, level_nama, created_at) value (?, ?, ?)', ['CUS', 'Pelanggan', now()]);
12         return 'insert data baru berhasil';
13     }
14 }
```

4. Kita coba jalankan di browser dengan url `localhost/PWL_POS/public/level` dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`

<div>← T →</div>				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	CUS	Pelanggan	2024-02-26 08:20:00	NULL



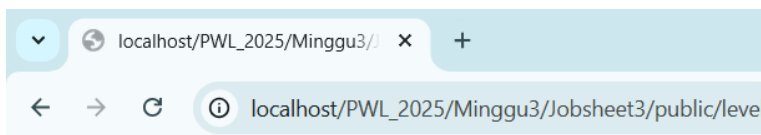
insert data baru berhasil

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	4	CUS	Pelanggan	2025-03-03 04:05:52	NULL

5. Selanjutnya, kita modifikasi lagi file `LevelController` untuk meng-*update* data di table `m_level` seperti berikut

```
LevelController.php x web.php
app > Http > Controllers > LevelController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17     }
18 }
```

6. Kita coba jalankan di browser dengan url `localhost/PWL_POS/public/level` lagi dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`



Update data berhasil. Jumlah data yang diupdate: 1 baris

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	4	CUS	Customer	2025-03-03 04:05:52	NULL



7. Kita coba modifikasi lagi file **LevelController** untuk melakukan proses hapus data

```
LevelController.php X web.php
app > Http > Controllers > LevelController.php > LevelController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17
18         $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
20     }
21 }
```

localhost/PWL_2025/Minggu3/ X +
localhost/PWL_2025/Minggu3/Jobsheet3/public/level

Delete data berhasil. Jumlah data yang dihapus: 1 baris

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL

8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table **m_level**. Kita modifikasi file **LevelController** seperti berikut

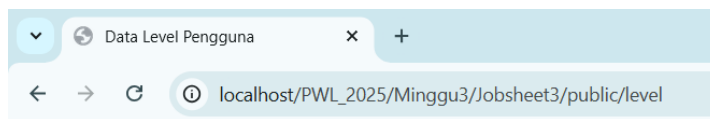
```
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17
18         // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
20
21         $data = DB::select('select * from m_level');
22         return view('level', ['data' => $data]);
23     }
24 }
```

9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil **view('level')**, maka kita buat file view pada VSCode di **PWL_POS/resources/view/level.blade.php**



```
resources > views > level.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Data Level Pengguna</title>
5   </head>
6   <body>
7     <h1>Data Level Pengguna</h1>
8     <table border="1" cellpadding="2" cellspacing="0">
9       <tr>
10        <th>ID</th>
11        <th>Kode Level</th>
12        <th>Nama Level</th>
13      </tr>
14      @foreach ($data as $d)
15        <tr>
16          <td>{{ $d->level_id }}</td>
17          <td>{{ $d->level_kode }}</td>
18          <td>{{ $d->level_nama }}</td>
19        </tr>
20      @endforeach
21    </table>
22  </body>
23 </html>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi



Data Level Pengguna

ID	Kode Level	Nama Level
1	ADM	Administrator
2	MNG	Manager
3	STF	Staff/Kasir

11. Laporkan hasil Praktikum-4 ini dan *commit* perubahan pada *git*.



E. QUERY BUILDER

Query builder adalah fitur yang disediakan Laravel untuk melakukan proses CRUD (*create, retrieve/read, update, delete*) pada database. Berbeda dengan *raw query* pada DB Facade yang mengharuskan kita menulis perintah SQL, pada *query builder* perintah SQL ini diakses menggunakan method. Jadi, kita tidak menulis perintah SQL secara langsung, melainkan cukup memanggil method-method yang ada di *query builder*.

Query builder membuat kode kita menjadi rapi dan lebih mudah dibaca. Selain itu *query builder* tidak terikat ke satu jenis database, jadi *query builder* bisa digunakan untuk mengakses berbagai jenis database seperti MySQL, MariaDB, PostgreSQL, SQL Server, dll. Jika suatu saat ingin beralih dari database MySQL ke PostgreSQL, tidak akan banyak kendala. Namun kelemahan dari *query builder* adalah kita harus mengetahui method-method apa saja yang ada di *query builder*.

INFO

Dokumentasi penggunaan Query Builder pada Laravel bisa dicek di laman ini

<https://laravel.com/docs/10.x/queries>

Ciri khas *query builder* Laravel adalah kita tentukan dahulu target table yang akan kita akses untuk operasi CRUD.

```
DB::table('<nama-tabel>'); // query builder untuk melakukan operasi CRUD pada tabel yang dituju
```

Perintah pertama yang dilakukan pada query builder adalah menentukan nama table yang akan dilakukan operasi CRUD. Kemudian baru disusul method yang ingin digunakan sesuai dengan peruntukannya. Contoh

- Perintah untuk *insert* data dengan method `insert()`

```
DB::table('m_kategori')->insert(['kategori_kode' => 'SMP', 'kategori_nama' => 'Smartphone']);
```

Query yang dihasilkan dari kode di atas adalah

```
insert into m_kategori(kategori_kode, kategori_nama) values('SMP', 'Smartphone');
```

- Perintah untuk *update* data dengan method `where()` dan `update()`

```
DB::table('m_kategori')->where('kategori_id', 1)->update(['kategori_nama' => 'Makanan Ringan']);
```

Query yang dihasilkan dari kode di atas adalah

```
update m_kategori set kategori_nama = 'Makanan Ringan' where kategori_id = 1;
```



- c. Perintah untuk *delete* data dengan method `where()` dan `delete()`

```
DB::table('m_kategori')->where('kategori_id', 9) ->delete();
```

Query yang dihasilkan dari kode di atas adalah

```
delete from m_kategori where kategori_id = 9;
```

- d. Perintah untuk ambil data

Method Query Builder	Query yang dihasilkan
<code>DB::table('m_kategori')->get();</code>	<code>select * from m_kategori</code>
<code>DB::table('m_kategori')->where('kategori_id', 1)->get();</code>	<code>select * from m_kategori where kategori_id = 1;</code>
<code>DB::table('m_kategori')->select('kategori_kode')->where('kategori_id', 1)->get();</code>	<code>select kategori_kode from m_kategori where kategori_id = 1;</code>

Praktikum 5 – Implementasi *Query Builder*

1. Kita buat controller dahulu untuk mengelola data pada table `m_kategori`

```
php artisan make:controller KategoriController
```

2. Kita modifikasi dulu untuk routing-nya, ada di `PWL_POS/routes/web.php`

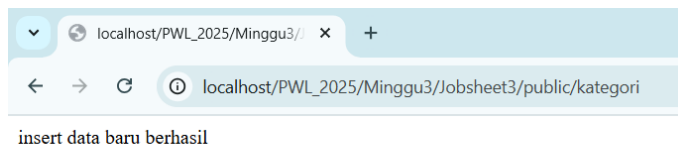
```
LevelController.php  KategoriController.php  level.blade.php  web.php X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use Illuminate\Support\Facades\Route;
6
7
8  Route::get('/', function () {
9      return view('welcome');
10 });
11
12 Route::get('/level', [LevelController::class, 'index']);
13 Route::get('/kategori', [KategoriController::class, 'index']);
```

3. Selanjutnya, kita modifikasi file `KategoriController` untuk menambahkan 1 data ke table `m_kategori`



```
LevelController.php KategoriController.php level.blade.php web.php
app > Http > Controllers > KategoriController.php > KategoriController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class KategoriController extends Controller
9 {
10     public function index()
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil';
19     }
20 }
```

4. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/kategori dan amati apa yang terjadi pada table `m_kategori` di database, *screenshot* perubahan yang ada pada table `m_kategori`



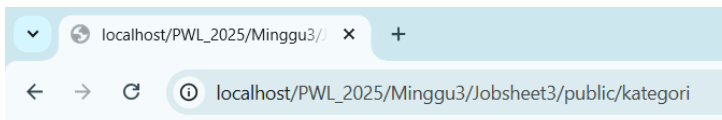
		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KT003	Snack	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KT004	Buah-buahan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KT005	Sayuran	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2025-03-03 04:44:04	NULL

5. Selanjutnya, kita modifikasi lagi file `KategoriController` untuk meng-*update* data di table `m_kategori` seperti berikut











```
app > Http > Controllers > KategoriController.php > KategoriController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     public function index()
11     {
12         /* $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil'; */
19
20         $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22     }
23 }
```

6. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/kategori lagi dan amati apa yang terjadi pada table `m_kategori` di database, *screenshot* perubahan yang ada pada table `m_kategori`

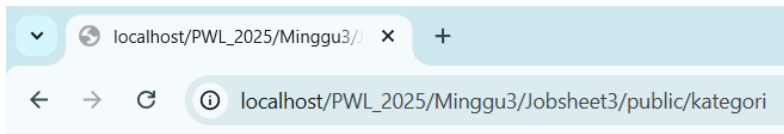


Update data berhasil. Jumlah data yang diupdate: 1 baris

<div>← ↩ →</div>				kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	KT001	Makanan	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	KT002	Pakaian	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	KT003	Snack	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	KT004	Buah-buahan	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	KT005	Sayuran	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	SNK	Camilan	2025-03-03 04:44:04	NULL

7. Kita coba modifikasi lagi file `KategoriController` untuk melakukan proses hapus data

```
10     public function index()
11     {
12         /* $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil'; */
19
20         // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22
23         $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24         return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
25     }
```



Delete data berhasil. Jumlah data yang dihapus: 1 baris

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KT003	Snack	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KT004	Buah-buahan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KT005	Sayuran	NULL	NULL

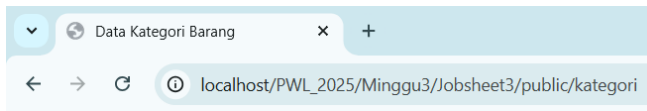
8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table `m_kategori`. Kita modifikasi file `KategoriController` seperti berikut

```
10 public function index()
11 {
12     /* $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17     DB::table('m_kategori')->insert($data);
18     return 'Insert data baru berhasil'; */
19
20     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21     // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22
23     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24     // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
25
26     $data = DB::table('m_kategori')->get();
27     return view('kategori', ['data' => $data]);
28 }
```

9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil `view('kategori')`, maka kita buat file view pada VSCode di `PWL_POS/resources/view/kategori.blade.php`

```
resources > views > kategori.blade.php > html > body > table > tr > td
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Data Kategori Barang</title>
5     </head>
6     <body>
7         <h1>Data Kategori Barang</h1>
8         <table border="1" cellpadding="2" cellspacing="0">
9             <tr>
10                 <th>ID</th>
11                 <th>Kode Kategori</th>
12                 <th>Nama Kategori</th>
13             </tr>
14             @foreach ($data as $d)
15                 <tr>
16                     <td>{{ $d->kategori_id }}</td>
17                     <td>{{ $d->kategori_kode }}</td>
18                     <td>{{ $d->kategori_nama }}</td>
19                 </tr>
20             @endforeach
21         </table>
22     </body>
23 </html>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi.



Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	KT001	Makanan
2	KT002	Pakaian
3	KT003	Snack
4	KT004	Buah-buahan
5	KT005	Sayuran

11. Laporkan hasil Praktikum-5 ini dan *commit* perubahan pada *git*

F. ELOQUENT ORM

Eloquent ORM adalah fitur bawaan dari laravel. Eloquent ORM adalah cara pengaksesan database dimana setiap baris tabel dianggap sebagai sebuah object. Kata ORM sendiri merupakan singkatan dari **Object-relational mapping**, yakni suatu teknik programming untuk mengkonversi data ke dalam bentuk object.

INFO

Eloquent ORM memerlukan Model untuk proses konversi data pada tabel menjadi object. Object inilah yang nantinya akan kita akses dari dalam controller. Oleh karena itu **membuat Model pada Laravel berarti menggunakan Eloquent ORM**. Silahkan cek disini

<https://laravel.com/docs/10.x/eloquent>

Perintah untuk membuat model adalah sebagai berikut

```
php artisan make:model <nama-model-CamelCase>
```

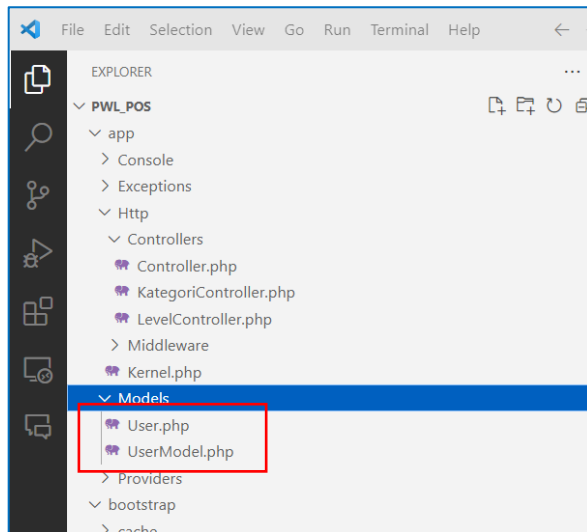
Untuk bisa melakukan operasi **CRUD** (*create, read/retrieve, update, delete*), kita harus membuat sebuah model sesuai dengan target tabel yang ingin digunakan. Jadi,

dalam 1 model, merepresentasikan 1 tabel database.

Praktikum 6 – Implementasi Eloquent ORM

1. Kita buat file model untuk tabel `m_user` dengan mengetikkan perintah

```
php artisan make:model UserModel
```



- Setelah berhasil generate model, terdapat 2 file pada folder `model` yaitu file `User.php` bawaan dari laravel dan file `UserModel.php` yang telah kita buat. Kali ini kita akan menggunakan file `UserModel.php`
- Kita buka file `UserModel.php` dan modifikasi seperti berikut

```
app > Models > UserModel.php > UserModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class UserModel extends Model
9  {
10     use HasFactory;
11
12     protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
13     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
14 }
15
```

```
app > Models > UserModel.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class UserModel extends Model
9  {
10     use HasFactory;
11
12     protected $table = 'm_user';
13     protected $primaryKey = 'user_id';
14 }
```




4. Kita modifikasi route `web.php` untuk mencoba routing ke controller `UserController`

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use App\Http\Controllers\UserController;
6  use Illuminate\Support\Facades\Route;
7
8
9  Route::get('/', function () {
10     return view('welcome');
11 });
12
13 Route::get('/level', [LevelController::class, 'index']);
14 Route::get('/kategori', [KategoriController::class, 'index']);
15 Route::get('/user', [UserController::class, 'index']);
```

```
Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
```

5. Sekarang, kita buat file controller `UserController` dan memodifikasinya seperti berikut

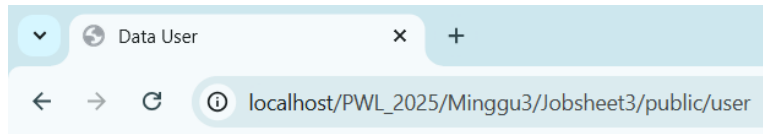
```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7
8  class UserController extends Controller
9  {
10     public function index()
11     {
12         // coba akses model UserModel
13         $user = UserModel::all(); // ambil semua data dari tabel m_user
14         return view('user', ['data' => $user]);
15     }
16 }
```

6. Kemudian kita buat view `user.blade.php`

```
resources > views > user.blade.php > ...
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Data User</title>
5      </head>
6      <body>
7          <h1>Data User</h1>
8          <table border="1" cellpadding="2" cellspacing="0">
9              <tr>
10                 <th>ID</th>
11                 <th>Username</th>
12                 <th>Nama</th>
13                 <th>ID Level Pengguna</th>
14             </tr>
15             @foreach ($data as $d)
16                 <tr>
17                     <td>{{ $d->user_id }}</td>
18                     <td>{{ $d->username }}</td>
19                     <td>{{ $d->nama }}</td>
20                     <td>{{ $d->level_id }}</td>
21                 </tr>
22             @endforeach
23         </table>
24     </body>
25 </html>
```



7. Jalankan di browser, catat dan laporkan apa yang terjadi



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3

8. Setelah itu, kita modifikasi lagi file `UserController`

```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserController extends Controller
10 {
11     public function index()
12     {
13         // tambah data user dengan Eloquent Model
14         $data = [
15             'username' => 'customer-1',
16             'nama' => 'Pelanggan',
17             'password' => Hash::make('12345'),
18             'level_id' => 4
19         ];
20         UserModel::insert($data); // tambahkan data ke tabel m_user
21
22         // coba akses model UserModel
23         $user = UserModel::all(); // ambil semua data dari tabel m_user
24         return view('user', ['data' => $user]);
25     }
26 }
```

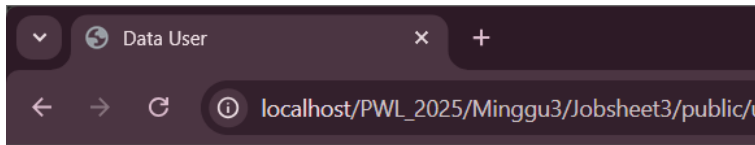
9. Jalankan di browser, amati dan laporkan apa yang terjadi

10. Kita modifikasi lagi file `UserController` menjadi seperti berikut

```
9  class UserController extends Controller
10 {
11     public function index()
12     {
13         // tambah data user dengan Eloquent Model
14         $data = [
15             'nama' => 'Pelanggan Pertama',
16         ];
17         UserModel::where('username', 'customer-1')->update($data); // update data user
18
19         // coba akses model UserModel
20         $user = UserModel::all(); // ambil semua data dari tabel m_user
21         return view('user', ['data' => $user]);
22     }
23 }
```



11. Jalankan di browser, amati dan laporkan apa yang terjadi



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
10	customer-1	Pelanggan Pertama	3

12. Jika sudah, laporkan hasil Praktikum-6 ini dan *commit* perubahan pada *git*

G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

- Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari **APP_KEY** pada *file setting .env* Laravel? APP_KEY digunakan untuk enkripsi data dalam Laravel, terutama untuk mengenkripsi session, password hashing, dan token keamanan lainnya.
- Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk **APP_KEY**? Menggunakan perintah pada terminal “ php atrisan key:generate
- Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
Secara default, Laravel memiliki tiga file migrasi:
 - create_users_table.php → Membuat tabel pengguna (users).
 - create_password_resets_table.php → Membuat tabel untuk menyimpan reset password.
 - create_failed_jobs_table.php → Menyimpan log pekerjaan (job) yang gagal dieksekusi.
- Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/output dari fungsi tersebut?
Digunakan untuk menambahkan dua kolom created_at dan updated_at untuk mencatat kapan data dibuat dan diperbarui.



5. Pada File Migrasi, terdapat fungsi `$table->id()`; Tipe data apa yang dihasilkan dari fungsi tersebut?
Tipe data yang dihasilkan oleh `$table->id()`;
Tipe data BIGINT (Unsigned Big Integer) dan menjadi primary key.
6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id()`; dengan menggunakan `$table->id('level_id')`; ?
Perbedaan dari `$table->id()`; dan `$table->id('level_id')`;
 - `$table->id()`; → Membuat primary key dengan nama default id.
 - `$table->id('level_id')`; → Membuat primary key dengan nama `level_id`.
7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
Untuk memastikan tidak terjadi duplikasi data
8. Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom `level_id` pada tabel `m_user` menggunakan `$tabel->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$tabel->id('level_id')` ?
 - `m_user`: `level_id` digunakan sebagai foreign key, sehingga harus bertipe `unsignedBigInteger`.
 - `m_level`: `level_id` dibuat dengan `$table->id('level_id')`;; yang default-nya `unsignedBigInteger` dan sebagai primary key.
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');` ?
 - Tujuan dari Class `Hash` yaitu digunakan untuk hashsing data (password)
 - Maksud dari kode `Hash::make('1234');`; yaitu digunakan mengubah String '1234' menjadi format hash yang aman.
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
Kegunaan tanda tanya(?) pada query builder yaitu sebagai placeholder untuk building parameter yang berguna untuk mencegah terjadinya SQL Injection



11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` ?
- `protected $table = 'm_user';` → Menentukan tabel yang digunakan oleh model.
 - `protected $primaryKey = 'user_id';` → Menentukan primary key dari tabel.
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan
- Menurut saya lebih mudah menggunakan Query Builder. Karena Menggunakan metode Laravel untuk membangun query, lebih mudah digunakan dibanding DB Façade

*** *Sekian, dan selamat belajar* ***