



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**

localhost/phpmyadmin/index.php?route=/server/databases

Server: localhost

Databases SQL Status User accounts Export Import

Databases

Create database PWL\_POS utf8mb4\_unicode\_ci Create

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**.

File Edit Selection View Go Run Terminal Help ← → ⌘ P WL\_POS

EXPLORER .env .env.example

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=
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=
```



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



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows a tree view with the current folder set to "PWL\_POS". The ".env.example" file is selected.
- Editor (Right):** Displays the contents of the ".env.example" file. Several environment variables are highlighted with red boxes:
  - APP\_KEY (Line 3)
  - DB\_CONNECTION (Line 11)
  - DB\_HOST (Line 12)
  - DB\_PORT (Line 13)
  - DB\_DATABASE (Line 14)
  - DB\_USERNAME (Line 15)
  - DB\_PASSWORD (Line 16)
- Status Bar:** Shows the path "PWL\_POS" and the file name ".env.example".

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

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:SSCqJWh1gb9ESYbgqtPhD2Wlo0vSr1UnP6YRXLU1c=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=pwl_pos
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

## B. MIGRATION

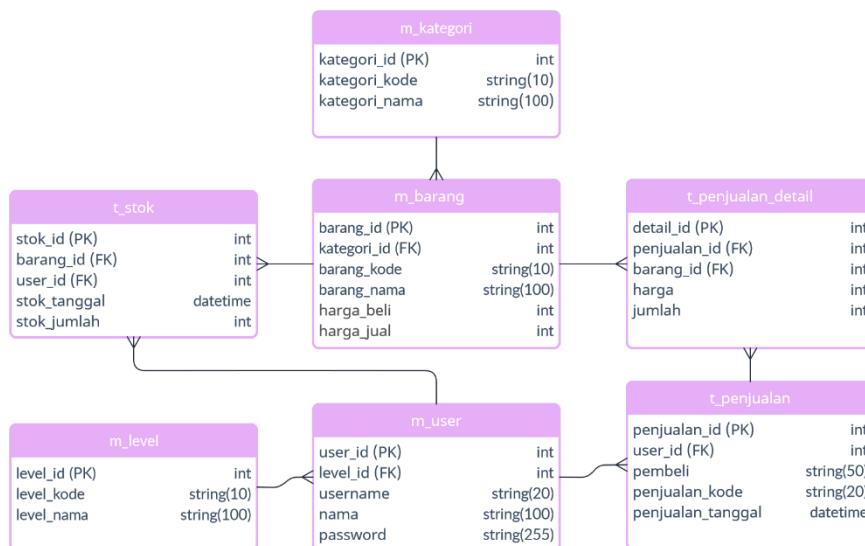
Migration pada Laravel merupakan sebuah fitur yang dapat membantu kita mengelola database secara efisien dengan menggunakan kode program. Migration membantu kita dalam membuat (*create*), mengubah (*edit*), dan menghapus (*delete*) struktur tabel dan kolom pada database yang sudah kita buat dengan cepat 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 dilanjut 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	<a href="#">m_level</a>	0
2	<a href="#">m_kategori</a>	0
3	<a href="#">m_user</a>	1
4	<a href="#">m_barang</a>	1
5	<a href="#">t_penjualan</a>	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

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

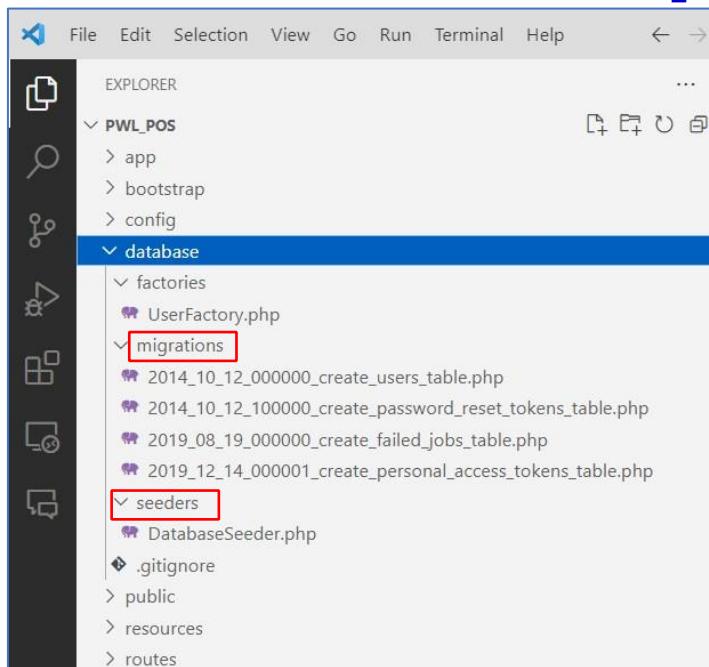
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- 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.

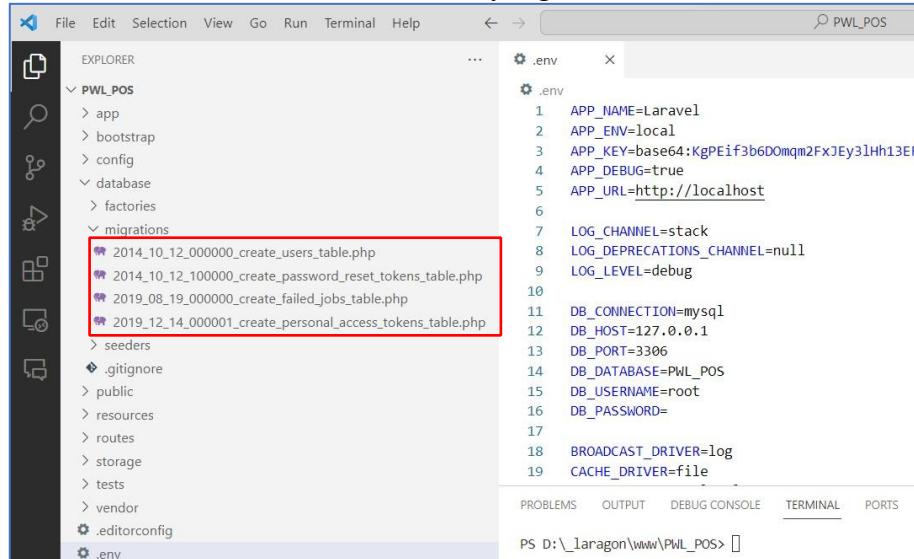
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder **PWL\_POS/database**



### Praktikum 2.1 - Pembuatan file migrasi tanpa relasi



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



```
File Edit Selection View Go Run Terminal Help
EXPLORER .env
PWL_POS
  app
  bootstrap
  config
  database
  factories
  migrations
    2014_10_12_000000_create_users_table.php
    2014_10_12_100000_create_password_reset_tokens_table.php
    2019_08_19_000000_create_failed_jobs_table.php
    2019_12_14_000001_create_personal_access_tokens_table.php
  seeders
  .gitignore
  public
  resources
  routes
  storage
  tests
  vendor
  .editorconfig
  .env
```

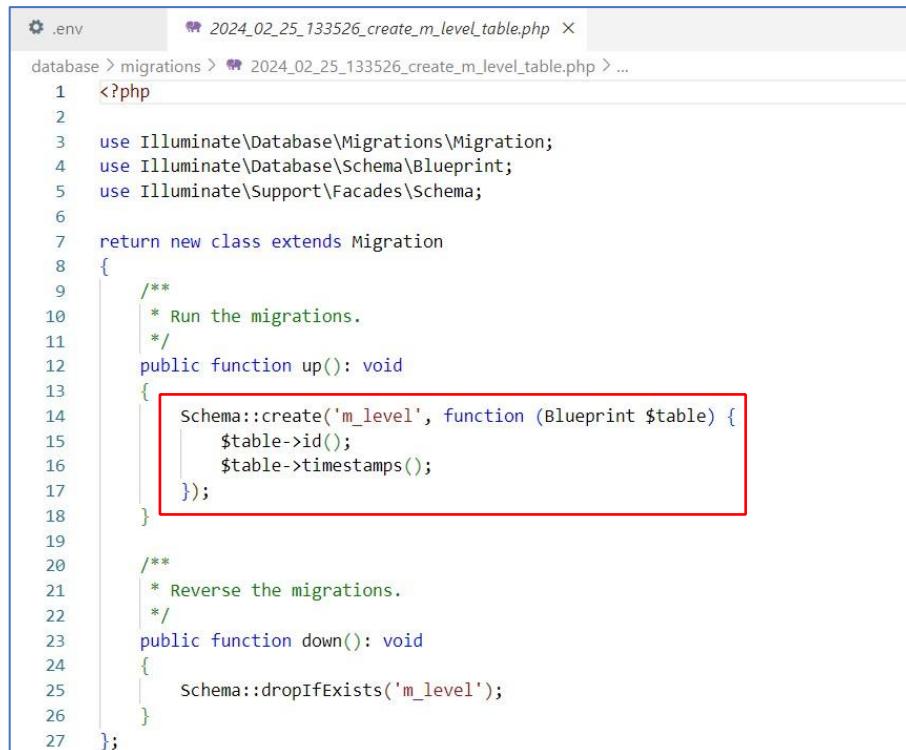
```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6DOMqm2FxJEy3lHh13EF
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATED_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=PWL_POS
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\\_laragon\www\PWL\_POS>

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
```



```
.env
2024_02_25_133526_create_m_level_table.php
```

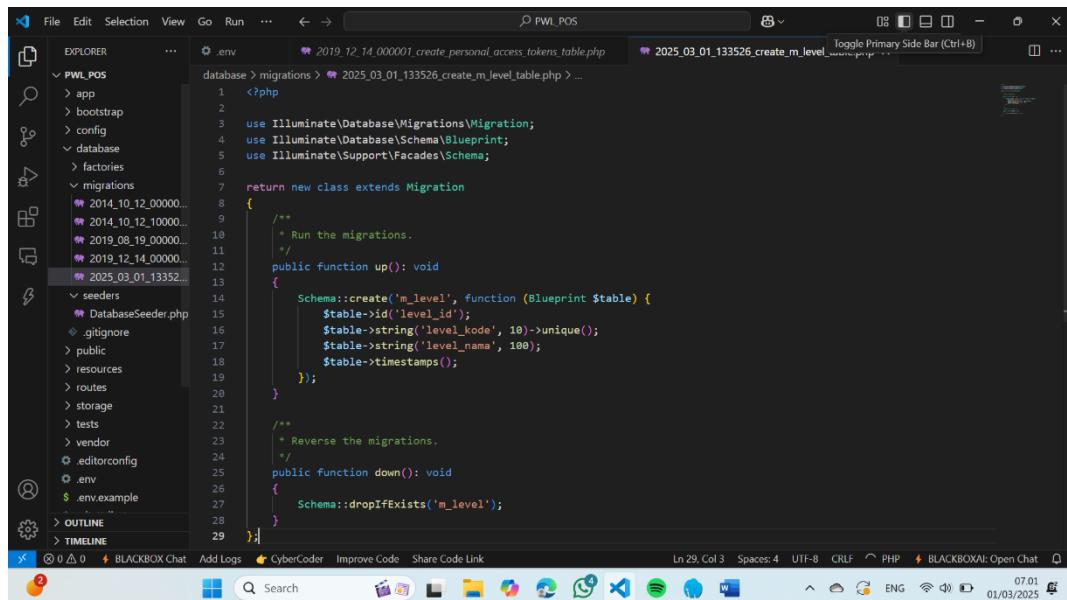
```
database > migrations > 2024_02_25_133526_create_m_level_table.php > ...
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('m_level', function (Blueprint $table) {
16             $table->id();
17             $table->timestamps();
18         });
19
20         /**
21          * Reverse the migrations.
22          */
23         public function down(): void
24         {
25             Schema::dropIfExists('m_level');
26         }
27     };
28 }
```

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 }
```

### Jawab: Membuat table m\_level



The screenshot shows a code editor window with a dark theme. The left sidebar displays a file tree for a Laravel project named 'PWL\_POS'. Under the 'database/migrations' directory, several migration files are listed, including '2014\_10\_12\_00000', '2014\_10\_12\_10000', '2019\_08\_19\_00000', '2019\_12\_14\_00000', and '2025\_03\_01\_13352'. The file '2025\_03\_01\_13352\_create\_m\_level\_table.php' is open in the main editor area. The code within this file is identical to the one shown in the previous code block, defining a migration for creating the 'm\_level' table with columns for 'level\_id', 'level\_kode', 'level\_nama', and 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
failed_jobs		0
migrations		5
<b>m_level</b>		0
password_reset_tokens		0
personal_access_tokens		0
users		0

### Jawab: Table tergenerate di phpMyAdmin

The screenshot shows the phpMyAdmin interface for the 'pwl\_pos' database. On the left, there's a tree view of databases and tables. In the main area, there's a search bar and a table listing tables with their details. The 'm\_level' table is highlighted with a red box. The table structure includes columns for id, level, and created\_at.

Table	Action	Rows	Type	Collation	Size	Overflow
failed_jobs		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	
migrations		5	InnoDB	utf8mb4_unicode_ci	16.0 Kib	
<b>m_level</b>		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	
password_reset_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	
personal_access_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	
users		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	
6 tables	Sum	5	InnoDB	utf8mb4_0900_ai_ci	96.0 Kib	

7. Ok, table sudah dibuat di database



8. Buat table *database* dengan *migration* untuk table **m\_kategori** yang sama-sama tidak memiliki *foreign key*

**Jawab: Buat table m\_kategori**

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** localhost:3306
- Database:** pwl\_pos
- Table:** m\_kategori
- Structure:** Shows the table structure with columns: kategori\_id, kategori\_kode, kategori\_nama, created\_at, and updated\_at.
- SQL:** A query is present: `SELECT * FROM `m_kategori`;`
- Operations:** Buttons for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, and Triggers.
- Query results operations:** Buttons for Create view and Console.

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**

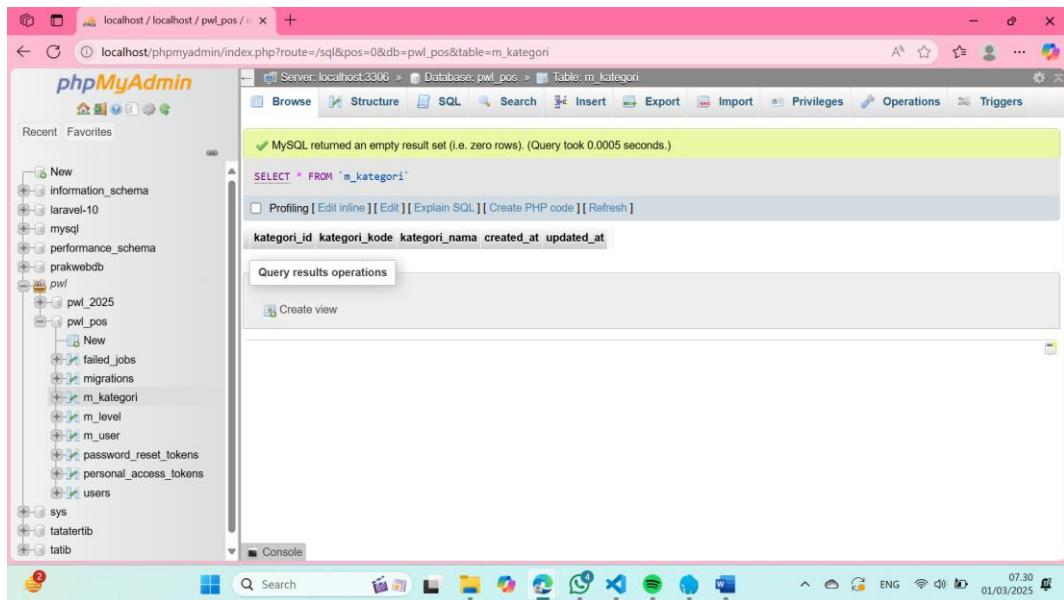
```
php artisan make:migration create_m_user_table --table=m_user
```

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 };
```

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

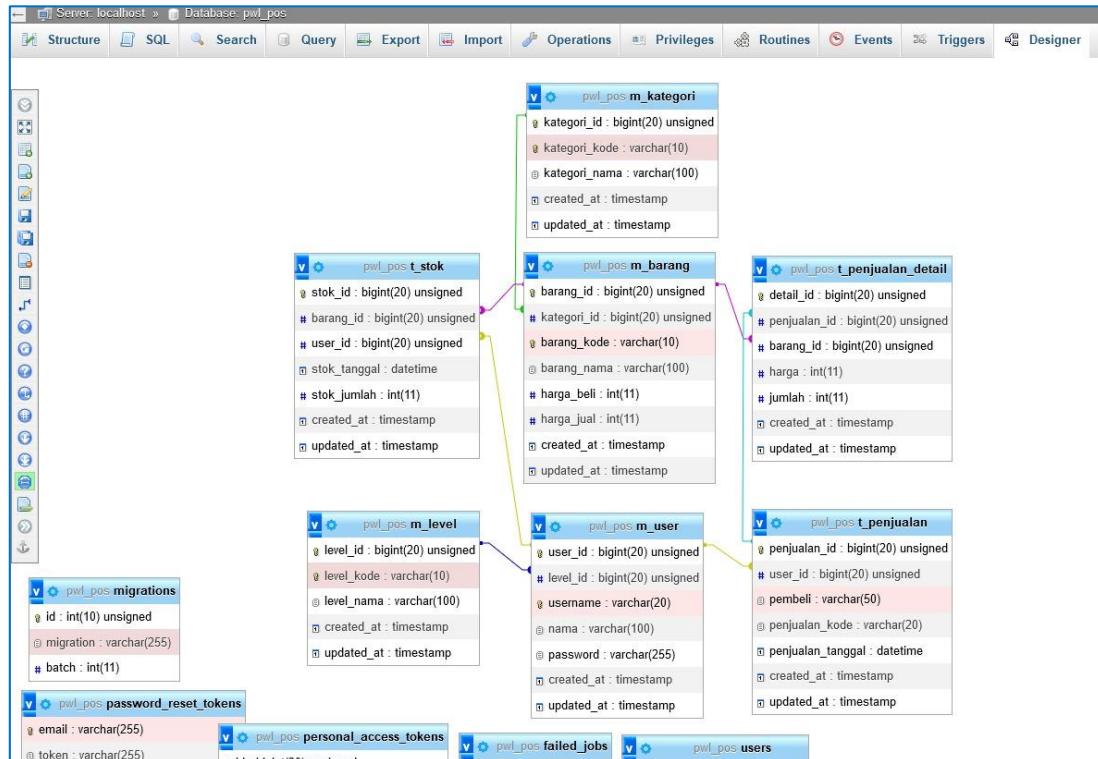
**Jawab:** Yang terjadi adalah table **m\_user** akan tertambahkan ke database



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

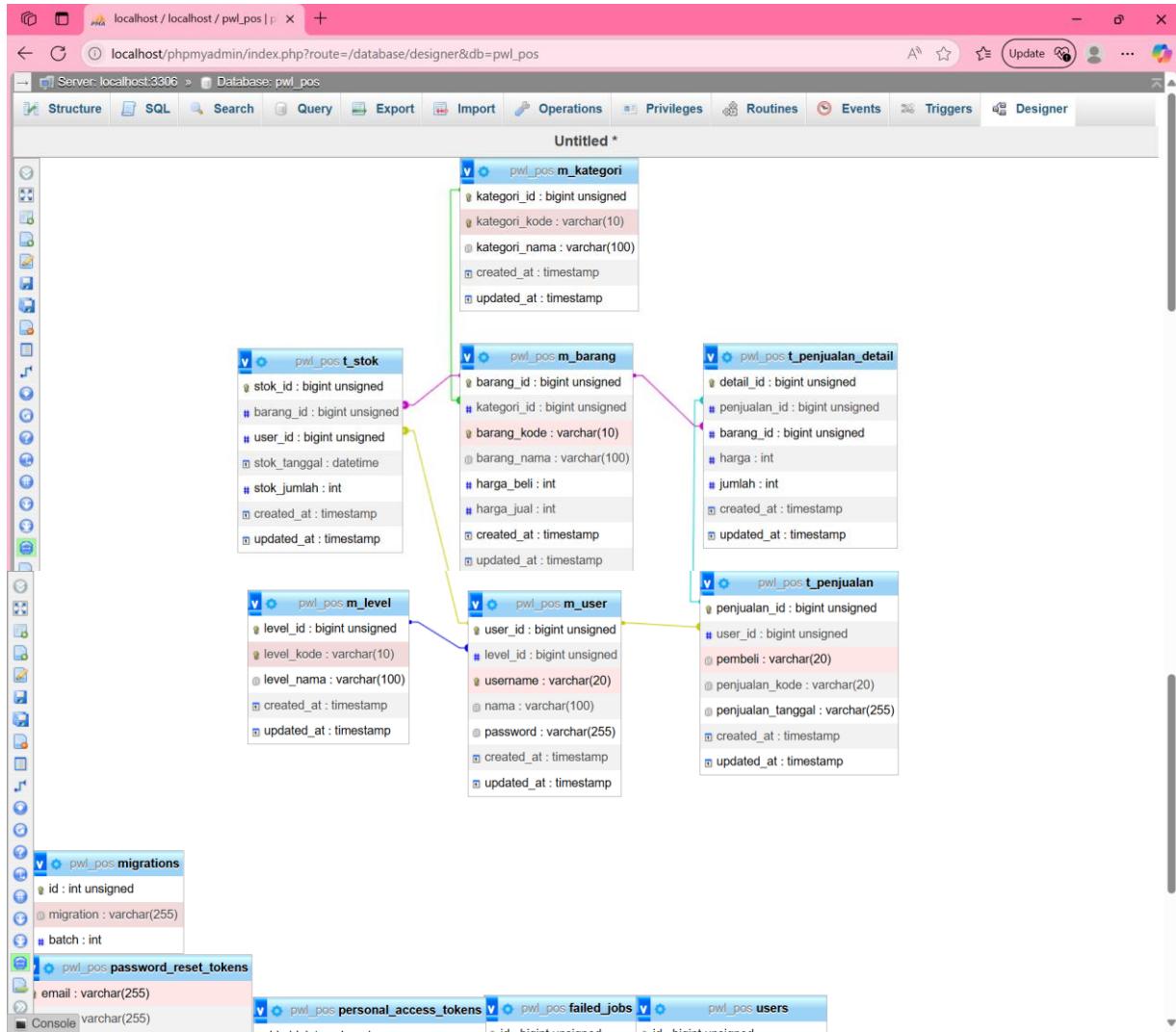
m_barang
t_penjualan
t_stok
t_penjualan_detail

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





**Jawab:**



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

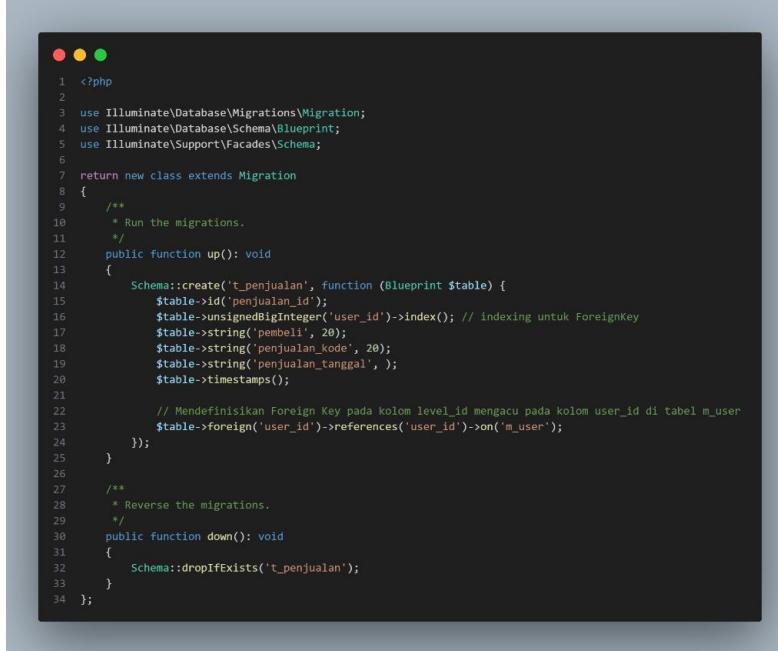


## m\_barang



```
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_barang', function (Blueprint $table) {
15             $table->id('barang_id');
16             $table->unsignedBigInteger('kategori_id')->index(); // indexing untuk ForeignKey
17             $table->string('barang_kode', 10)->unique();
18             $table->string('barang_nama', 100);
19             $table->integer('harga_beli');
20             $table->integer('harga_jual');
21             $table->timestamps();
22
23             $table->foreign('kategori_id')->references('kategori_id')->on('m_kategori');
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('m_barang');
33     }
34 };
```

## t\_penjualan



```
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('t_penjualan', function (Blueprint $table) {
15             $table->id('penjualan_id');
16             $table->unsignedBigInteger('user_id')->index(); // indexing untuk ForeignKey
17             $table->string('pembeli', 20);
18             $table->string('penjualan_kode', 20);
19             $table->string('penjualan_tanggal', );
20             $table->timestamps();
21
22             // Mendefinisikan Foreign Key pada kolom level_id mengacu pada kolom user_id di tabel m_user
23             $table->foreign('user_id')->references('user_id')->on('m_user');
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('t_penjualan');
33     }
34 };
```



## t\_stok

```
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('t_stok', function (Blueprint $table) {
15             $table->id('stok_id');
16             $table->unsignedBigInteger('barang_id')->index();
17             $table->unsignedBigInteger('user_id')->index();
18             $table->datetime('stok_tanggal');
19             $table->integer('stok_jumlah');
20             $table->timestamps();
21
22             // foreignkey
23             $table->foreign('barang_id')->references('barang_id')->on('m_barang');
24             $table->foreign('user_id')->references('user_id')->on('m_user');
25
26         });
27     }
28
29     /**
30      * Reverse the migrations.
31      */
32     public function down(): void
33     {
34         Schema::dropIfExists('t_stok');
35     }
36 };
```

## t\_penjualan\_detail

```
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('t_penjualan_detail', function (Blueprint $table) {
15             $table->bigIncrements('detail_id');
16             $table->unsignedBigInteger('penjualan_id')->index(); // indexing untuk ForeignKey
17             $table->unsignedBigInteger('barang_id')->index(); // indexing untuk ForeignKey
18             $table->integer('harga');
19             $table->integer('jumlah');
20             $table->timestamps(); // otomatis membuat kolom created_at dan updated_at
21
22             // Mendefinisikan Foreign Key
23             $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan');
24             $table->foreign('barang_id')->references('barang_id')->on('m_barang');
25
26         });
27     }
28
29     /**
30      * Reverse the migrations.
31      */
32     public function down(): void
33     {
34         Schema::dropIfExists('t_penjualan_detail');
35     }
36 };
```



## 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>
```

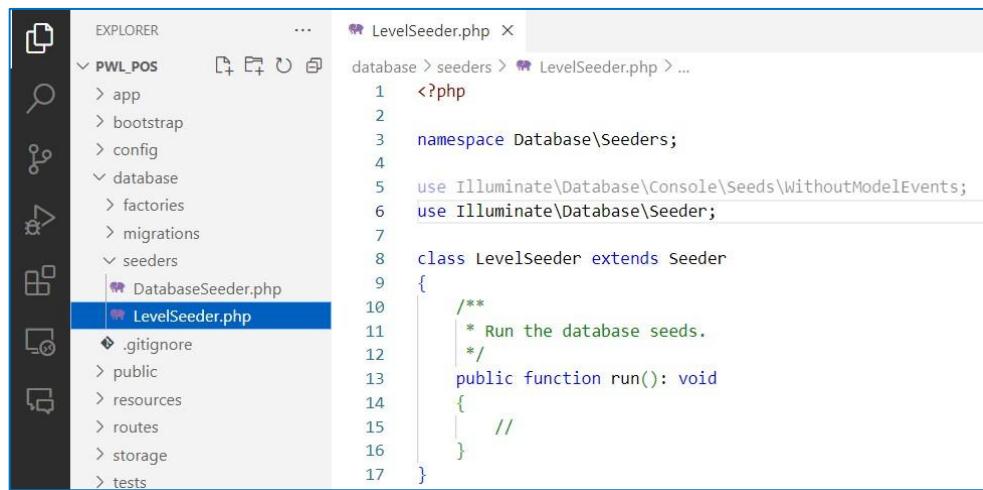
Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau *dummy* data untuk memudahkan pengujian dan pengembangan aplikasi kita. Sehingga fitur *seeder* bisa kita pakai dalam membuat sebuah aplikasi web.



### Praktikum 3 – Membuat file seeder

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

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

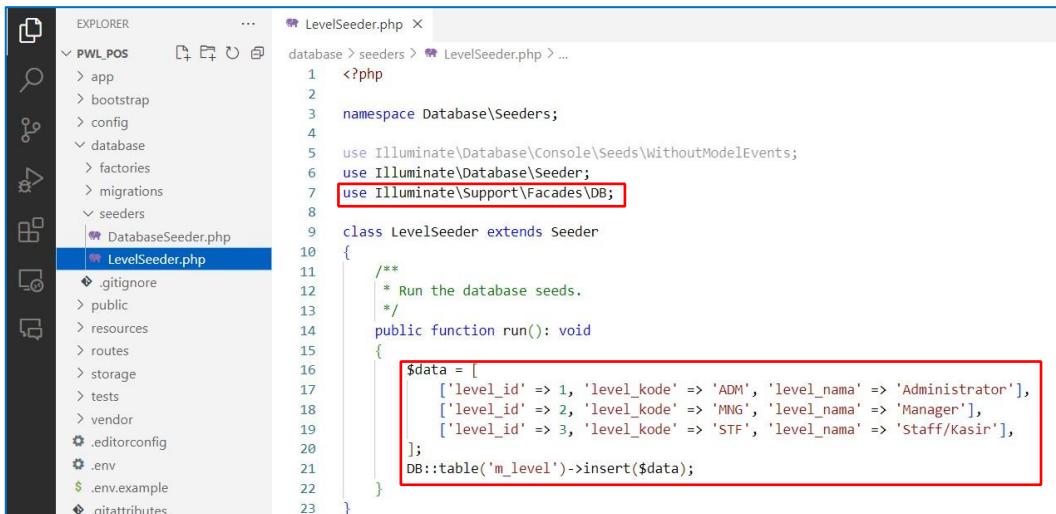


The screenshot shows a code editor interface with a sidebar labeled "EXPLORER". In the sidebar, under the "seeders" folder, a new file named "LevelSeeder.php" is being created. The main pane displays the generated PHP code for a Seeder:

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 class LevelSeeder extends Seeder
9 {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         //
16     }
17 }
```



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



```
database > seeders > LevelSeeder.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 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

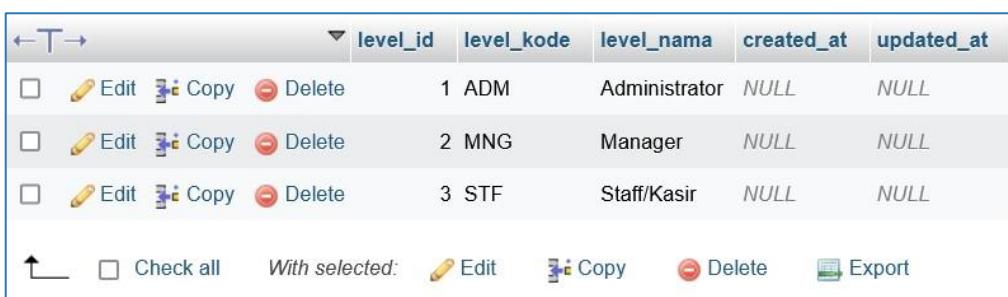
```
php artisan db:seed --class=LevelSeeder
```



```
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



### Jawab: seeder m\_level

The screenshot shows the phpMyAdmin interface for the 'pwl\_pos' database. The left sidebar lists tables such as prakwebdb, pwl, pwl\_2025, pwl\_pos, failed\_jobs, migrations, m\_barang, m\_kategori, m\_level, m\_user, password\_reset\_tokens, personal\_access\_tokens, t\_penjualan, t\_penjualan\_detail, t\_stok, and users. The 'm\_level' table is selected in the center pane. The table structure includes columns: level\_id, level\_kode, level\_nama, created\_at, and updated\_at. The data grid shows three rows:

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	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 }
```

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"/>		1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwY.4oAKU7FzwS6fXV...
<input type="checkbox"/>		2	2	manager	Manager	\$2y\$12\$Ajfnz20/FdPTeUgghz31muEhlFaruLxkh5wvZ9NGRp...
<input type="checkbox"/>		3	3	staff	Staff/Kasir	\$2y\$12\$Gj23TqGclW5pYeR0VL4o5OxPwb3OsK99VMy/BHnbJ9W...



### Jawab: seeder m\_user

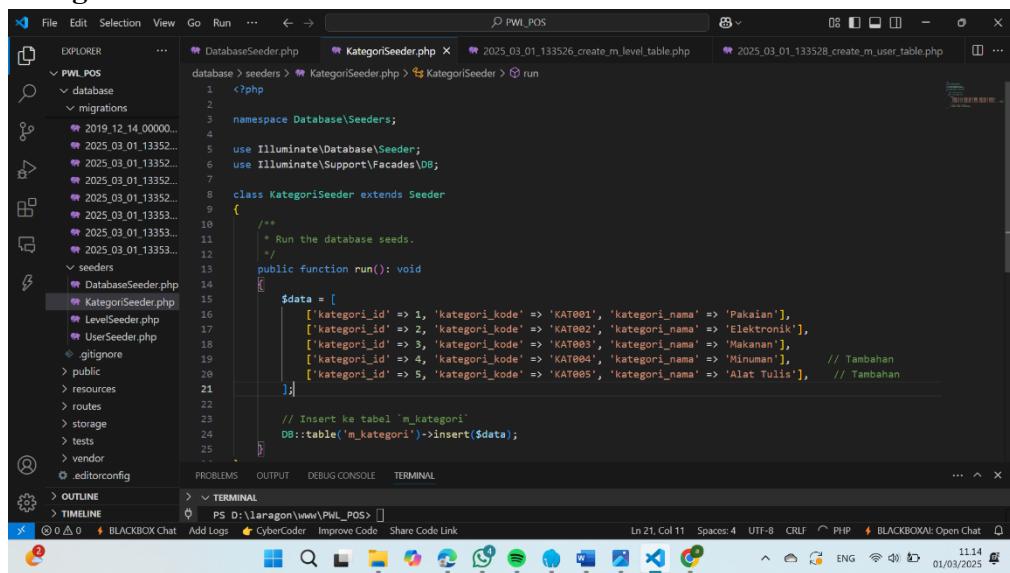
	user_id	level_id	username	nama	password	created_at
<input type="checkbox"/>	1	1	admin	Administrator	\$2y\$12\$AzJ4oZBFgMnfU6n.ca5EuKhA2QGJS9M.bOrBd.cRg...	NULL
<input type="checkbox"/>	2	2	Manager	Manager	\$2y\$12\$JUG3zqpQOGCzad8SpLDKK.c6jZolwBcbWNTYZp3YQA7...	NULL
<input type="checkbox"/>	3	3	Staff	Staff/Kasir	\$2y\$12\$NjHWx66Z3k.X60kg70jXI.dLT3H9OHoXZnCs4GbSmkc...	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	<a href="#">m_kategori</a>	5	5 kategori barang
2	<a href="#">m_barang</a>	10	10 barang yang berbeda
3	<a href="#">t_stok</a>	10	Stok untuk 10 barang
4	<a href="#">t_penjualan</a>	10	10 transaksi penjualan
5	<a href="#">t_penjualan_detail</a>	30	3 barang untuk setiap transaksi penjualan



## KategoriSeeder



```
<?php
namespace Database\Seeders;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
class KategoriSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['kategori_id' => 1, 'kategori.kode' => 'KAT001', 'kategori.nama' => 'Pakaian'],
            ['kategori_id' => 2, 'kategori.kode' => 'KAT002', 'kategori.nama' => 'Elektronik'],
            ['kategori_id' => 3, 'kategori.kode' => 'KAT003', 'kategori.nama' => 'Makanan'],
            ['kategori_id' => 4, 'kategori.kode' => 'KAT004', 'kategori.nama' => 'Minuman'], // Tambahan
            ['kategori_id' => 5, 'kategori.kode' => 'KAT005', 'kategori.nama' => 'Alat Tulis'], // Tambahan
        ];
        DB::table('m_kategori')->insert($data);
    }
}
```



	kategori_id	kategori.kode	kategori.nama	created_at	updated_at
<input type="checkbox"/>	1	KAT001	Pakaian	NULL	NULL
<input type="checkbox"/>	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	3	KAT003	Makanan	NULL	NULL
<input type="checkbox"/>	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	5	KAT005	Alat Tulis	NULL	NULL



## BarangSeeder

```
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 BarangSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             [
18                 'barang_id' => 1,
19                 'barang_kode' => 'BRG001',
20                 'barang_nama' => 'Laptop',
21                 'harga_beli' => 5000000,
22                 'harga_jual' => 7000000,
23                 'kategori_id' => 1,
24             ],
25             [
26                 'barang_id' => 2,
27                 'barang_kode' => 'BRG002',
28                 'barang_nama' => 'Smartphone',
29                 'harga_beli' => 3000000,
30                 'harga_jual' => 4000000,
31                 'kategori_id' => 1,
32             ],
33             [
34                 'barang_id' => 3,
35                 'barang_kode' => 'BRG003',
36                 'barang_nama' => 'T-Shirt',
37                 'harga_beli' => 50000,
38                 'harga_jual' => 100000,
39                 'kategori_id' => 2,
40             ],
41             [
42                 'barang_id' => 4,
43                 'barang_kode' => 'BRG004',
44                 'barang_nama' => 'Jeans',
45                 'harga_beli' => 100000,
46                 'harga_jual' => 200000,
47                 'kategori_id' => 2,
48             ],
49             [
50                 'barang_id' => 5,
51                 'barang_kode' => 'BRG005',
52                 'barang_nama' => 'Pensil',
53                 'harga_beli' => 2000,
54                 'harga_jual' => 5000,
55                 'kategori_id' => 3,
56             ],
57             [
58                 'barang_id' => 6,
59                 'barang_kode' => 'BRG006',
60                 'barang_nama' => 'Buku Tulis',
61                 'harga_beli' => 5000,
62                 'harga_jual' => 10000,
63                 'kategori_id' => 3,
64             ],
65             [
66                 'barang_id' => 7,
67                 'barang_kode' => 'BRG007',
68                 'barang_nama' => 'Kerupuk',
69                 'harga_beli' => 1000,
70                 'harga_jual' => 2000,
71                 'kategori_id' => 4,
72             ],
73             [
74                 'barang_id' => 8,
75                 'barang_kode' => 'BRG008',
76                 'barang_nama' => 'Nasi',
77                 'harga_beli' => 5000,
78                 'harga_jual' => 10000,
79                 'kategori_id' => 4,
80             ],
81             [
82                 'barang_id' => 9,
83                 'barang_kode' => 'BRG009',
84                 'barang_nama' => 'Air Mineral',
85                 'harga_beli' => 3000,
86                 'harga_jual' => 5000,
87                 'kategori_id' => 5,
88             ],
89             [
90                 'barang_id' => 10,
91                 'barang_kode' => 'BRG010',
92                 'barang_nama' => 'Soda',
93                 'harga_beli' => 5000,
94                 'harga_jual' => 8000,
95                 'kategori_id' => 5,
96             ],
97         ];
98         DB::table('m_barang')->insert($data);
99     }
100 }
```



## StokSeeder

```
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 use function Laravel\Prompts\table;
10
11 class StokSeeder extends Seeder
12 {
13     /**
14      * Run the database seeds.
15     */
16     public function run(): void
17     {
18         $data = [
19             [
20                 'stok_id' => 1,
21                 'stok_jumlah' => 100,
22                 'stok_tanggal' => '2024-02-27 08:00:00',
23                 'barang_id' => 1,
24                 'user_id' => 1,
25             ],
26             [
27                 'stok_id' => 2,
28                 'stok_jumlah' => 120,
29                 'stok_tanggal' => '2024-02-27 08:00:00',
30                 'barang_id' => 2,
31                 'user_id' => 1,
32             ],
33             [
34                 'stok_id' => 3,
35                 'stok_jumlah' => 10,
36                 'stok_tanggal' => '2024-02-27 08:00:00',
37                 'barang_id' => 3,
38                 'user_id' => 1,
39             ],
40             [
41                 'stok_id' => 4,
42                 'stok_jumlah' => 100,
43                 'stok_tanggal' => '2024-02-27 08:00:00',
44                 'barang_id' => 4,
45                 'user_id' => 2,
46             ],
47             [
48                 'stok_id' => 5,
49                 'stok_jumlah' => 100,
50                 'stok_tanggal' => '2024-02-27 08:00:00',
51                 'barang_id' => 5,
52                 'user_id' => 2,
53             ],
54             [
55                 'stok_id' => 6,
56                 'stok_jumlah' => 100,
57                 'stok_tanggal' => '2024-02-27 08:00:00',
58                 'barang_id' => 6,
59                 'user_id' => 2,
60             ],
61             [
62                 'stok_id' => 7,
63                 'stok_jumlah' => 100,
64                 'stok_tanggal' => '2024-02-27 08:00:00',
65                 'barang_id' => 7,
66                 'user_id' => 3,
67             ],
68             [
69                 'stok_id' => 8,
70                 'stok_jumlah' => 100,
71                 'stok_tanggal' => '2024-02-27 08:00:00',
72                 'barang_id' => 8,
73                 'user_id' => 3,
74             ],
75             [
76                 'stok_id' => 9,
77                 'stok_jumlah' => 100,
78                 'stok_tanggal' => '2024-02-27 08:00:00',
79                 'barang_id' => 9,
80                 'user_id' => 3,
81             ],
82             [
83                 'stok_id' => 10,
84                 'stok_jumlah' => 100,
85                 'stok_tanggal' => '2024-02-27 08:00:00',
86                 'barang_id' => 10,
87                 'user_id' => 1,
88             ],
89         ];
90         DB::table('t_stok')->insert($data);
91     }
92 }
```

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	1	1	1	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	2	2	1	2024-02-27 08:00:00	120	NULL	NULL
<input type="checkbox"/>	3	3	1	2024-02-27 08:00:00	10	NULL	NULL
<input type="checkbox"/>	4	4	2	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	5	5	2	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	6	6	2	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	7	7	3	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	8	8	3	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	9	9	3	2024-02-27 08:00:00	100	NULL	NULL
<input type="checkbox"/>	10	10	1	2024-02-27 08:00:00	100	NULL	NULL



## PenjualanSeeder

```
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     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             [
18                 'penjualan_id' => 1,
19                 'pembeli' => 'John Doe',
20                 'penjualan_kode' => 'PJM001',
21                 'penjualan_tanggal' => '2024-02-27 08:00:00',
22                 'user_id' => 1,
23             ],
24             [
25                 'penjualan_id' => 2,
26                 'pembeli' => 'Jane Smith',
27                 'penjualan_kode' => 'PJM002',
28                 'penjualan_tanggal' => '2024-02-27 08:30:00',
29                 'user_id' => 1,
30             ],
31             [
32                 'penjualan_id' => 3,
33                 'pembeli' => 'Michael Johnson',
34                 'penjualan_kode' => 'PJM003',
35                 'penjualan_tanggal' => '2024-02-27 09:00:00',
36                 'user_id' => 1,
37             ],
38             [
39                 'penjualan_id' => 4,
40                 'pembeli' => 'Emily Brown',
41                 'penjualan_kode' => 'PJM004',
42                 'penjualan_tanggal' => '2024-02-27 09:30:00',
43                 'user_id' => 1,
44             ],
45             [
46                 'penjualan_id' => 5,
47                 'pembeli' => 'David Wilson',
48                 'penjualan_kode' => 'PJM005',
49                 'penjualan_tanggal' => '2024-02-27 08:00:00',
50                 'user_id' => 2,
51             ],
52             [
53                 'penjualan_id' => 6,
54                 'pembeli' => 'Jessica Lee',
55                 'penjualan_kode' => 'PJM006',
56                 'penjualan_tanggal' => '2024-02-27 08:30:00',
57                 'user_id' => 2,
58             ],
59             [
60                 'penjualan_id' => 7,
61                 'pembeli' => 'Christopher Taylor',
62                 'penjualan_kode' => 'PJM007',
63                 'penjualan_tanggal' => '2024-02-27 08:00:00',
64                 'user_id' => 2,
65             ],
66             [
67                 'penjualan_id' => 8,
68                 'pembeli' => 'Sarah Martinez',
69                 'penjualan_kode' => 'PJM008',
70                 'penjualan_tanggal' => '2024-02-27 08:30:00',
71                 'user_id' => 3,
72             ],
73             [
74                 'penjualan_id' => 9,
75                 'pembeli' => 'Daniel Garcia',
76                 'penjualan_kode' => 'PJM009',
77                 'penjualan_tanggal' => '2024-02-27 08:00:00',
78                 'user_id' => 3,
79             ],
80             [
81                 'penjualan_id' => 10,
82                 'pembeli' => 'Olivia Rodriguez',
83                 'penjualan_kode' => 'PJM010',
84                 'penjualan_tanggal' => '2024-02-27 08:30:00',
85                 'user_id' => 3,
86             ],
87         ];
88         DB::table('t_penjualan')->insert($data);
89     }
90 }
```

Extra options							
	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	1	John Doe	PJM001	2024-02-27 08:00:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	1	Jane Smith	PJM002	2024-02-27 08:30:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	1	Michael Johnson	PJM003	2024-02-27 09:00:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	1	Emily Brown	PJM004	2024-02-27 09:30:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	2	David Wilson	PJM005	2024-02-27 08:00:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	6	2	Jessica Lee	PJM006	2024-02-27 08:30:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	7	2	Christopher Taylor	PJM007	2024-02-27 08:00:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	8	3	Sarah Martinez	PJM008	2024-02-27 08:30:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	9	3	Daniel Garcia	PJM009	2024-02-27 08:00:00	NULL	NULL
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	10	3	Olivia Rodriguez	PJM010	2024-02-27 08:30:00	NULL	NULL



## PenjualanDetailSeeder

```
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 PenjualanDetailSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [];
17
18         // Menentukan jumlah transaksi penjualan yang diinginkan
19         $jumlah_transaksi = 10;
20
21         // Menambahkan setiap transaksi penjualan dengan 3 barang
22         for ($i = 1; $i <= $jumlah_transaksi; $i++) {
23             $barang_ids = range(1, 3); // Mengvertakan 3 barang dalam setiap transaksi
24             foreach ($barang_ids as $barang_id) {
25                 $data[] = [
26                     'detail_id' => count($data) + 1,
27                     'harga' => rand(100000, 2000000), // Harga acak untuk setiap barang
28                     'jumlah' => rand(1, 5), // Jumlah acak untuk setiap barang
29                     'penjualan_id' => $i,
30                     'barang_id' => $barang_id,
31                 ];
32             }
33         }
34
35         DB::table('t_penjualan_detail')->insert($data);
36     }
37 }
```

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	1	1	10647875	2	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	1	2	7011849	3	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	1	3	248460	3	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	2	1	18926944	2	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	2	2	17452130	1	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	6	2	3	14807038	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	7	3	1	11706058	2	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	8	3	2	14349462	3	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	9	3	3	8152956	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	10	4	1	17533231	5	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	11	4	2	19426993	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	12	4	3	2766657	1	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	13	5	1	11809987	5	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	14	5	2	18492004	5	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	15	5	3	7618855	1	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	16	6	1	19009651	3	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	17	6	2	7604564	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	18	6	3	18456353	1	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	19	7	1	7162480	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	20	7	2	6231170	5	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	21	7	3	10509109	5	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	22	8	1	10277735	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	23	8	2	10487679	1	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	24	8	3	9546526	4	NULL	NULL
<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	25	9	1	11492876	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`

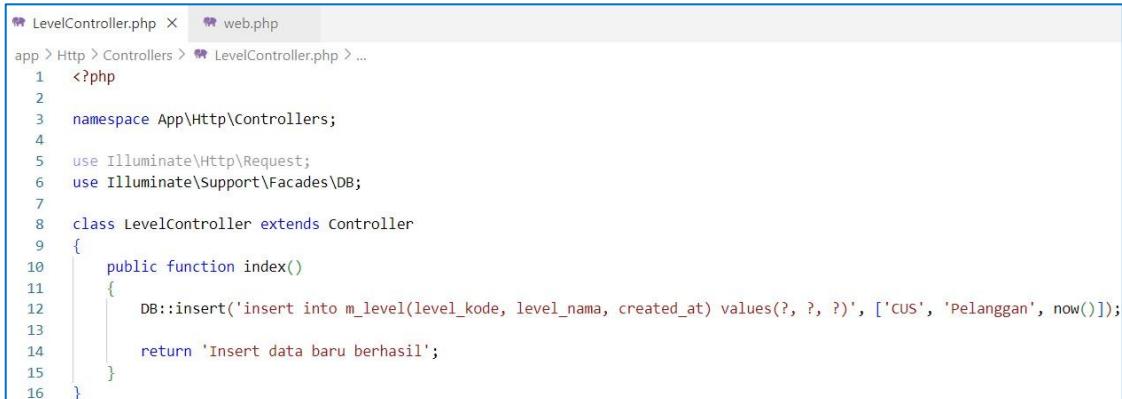
```
php artisan make:controller LevelController
```

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

```
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**



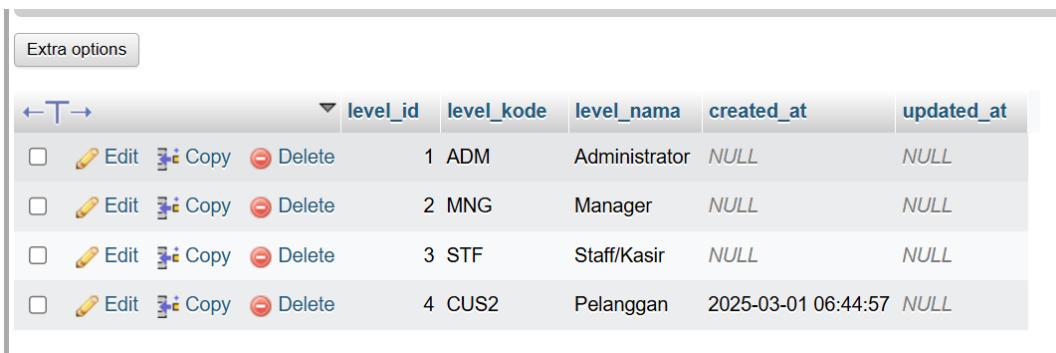
```
LevelController.php × web.php
app > Http > Controllers > LevelController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         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 }
```

4. Kita coba jalankan di browser dengan url [localhost/PWL\\_POS/public/level](http://localhost/PWL_POS/public/level) dan amati apa yang terjadi pada table **m\_level** di database, screenshot perubahan yang ada pada table **m\_level**



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

### Jawab



	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	4	CUS2	Pelanggan	2025-03-01 06:44:57	NULL



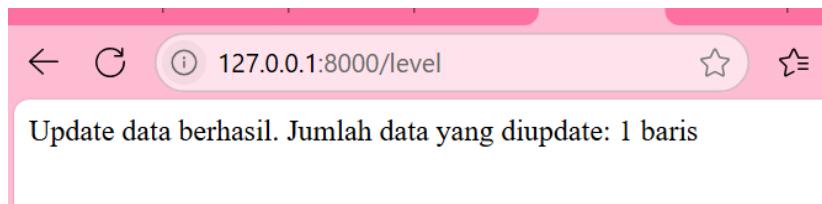
5. Selanjutnya, kita modifikasi lagi file `LevelController` untuk meng-update data di table `m_level` seperti berikut

```
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`

**Jawab:**

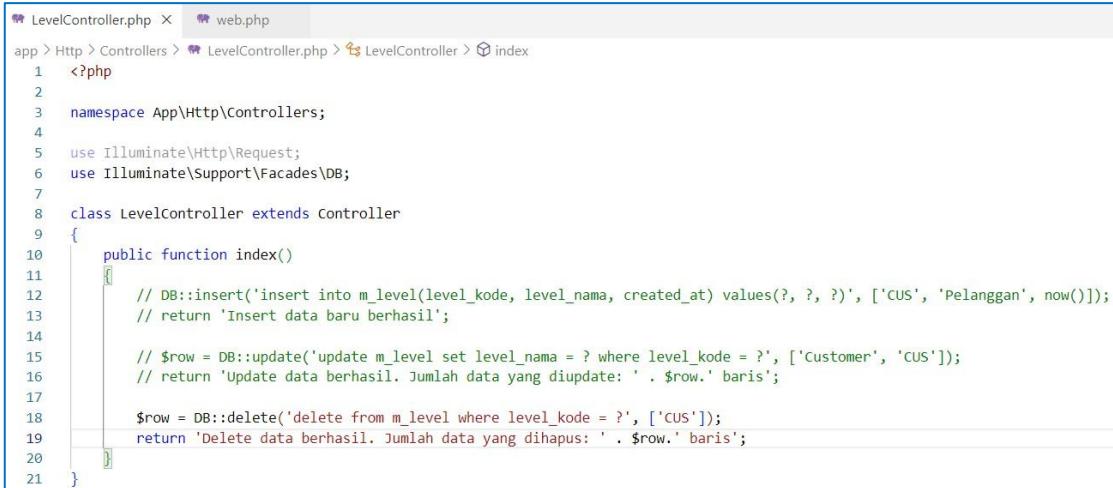
Perubahan yang terjadi di url dan database yang awalnya level CUS2 dengan nama Pelanggan dapat di update menjadi customer



Extra options						
	← T →	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	<a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	CUS2	Customer	2025-03-01 06:44:57	NULL

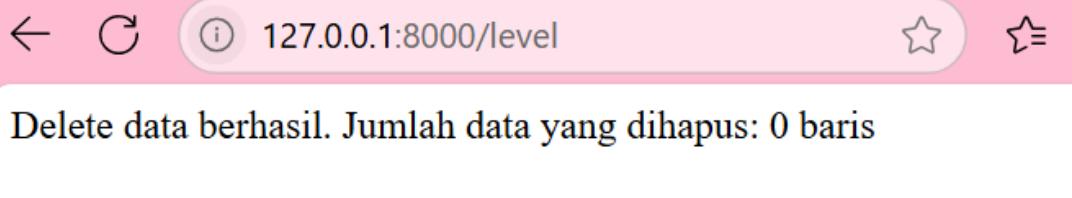


7. Kita coba modifikasi lagi file **LevelController** untuk melakukan proses hapus data



```
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 }
```

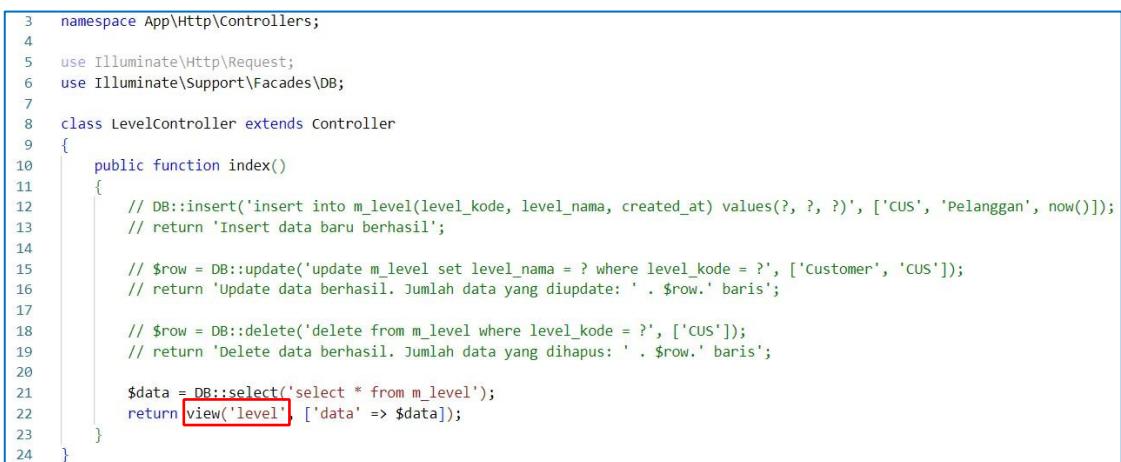
**Jawab**



← ⌂ ⓘ 127.0.0.1:8000/level ⭐ ⌂ ⓘ

Delete data berhasil. Jumlah data yang dihapus: 0 baris

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

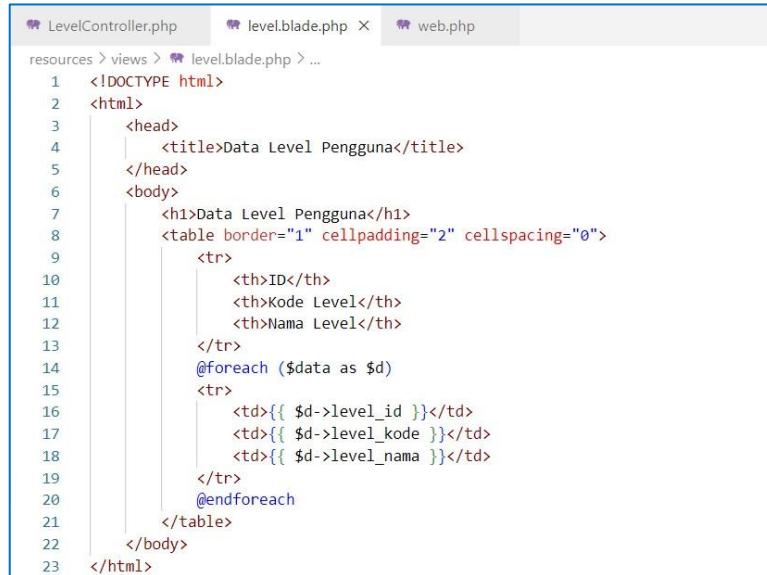


```
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['cus', '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`

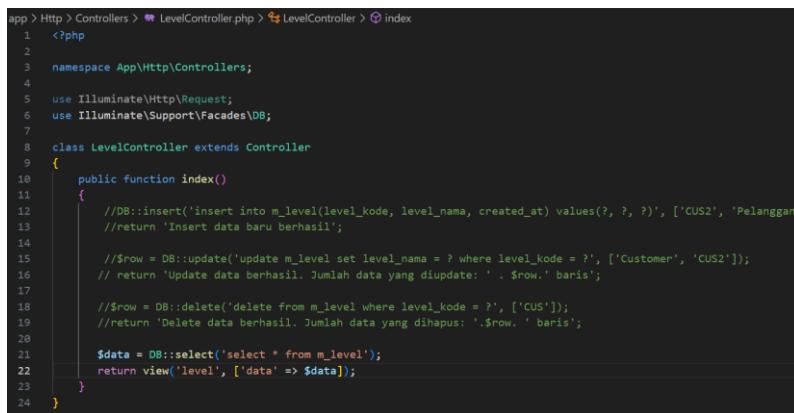


```
resources > views > level.blade.php > ...
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Data Level Pengguna</title>
5      </head>
6      <body>
7          <h1>Data Level Pengguna</h1>
8          <table border="1" cellpadding="2" cellspacing="0">
9              <tr>
10                 <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

### Jawab

LeverController



```
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(?, ?, ?)', ['CUS2', 'Pelanggan']);
13         //return 'Insert data baru berhasil';
14
15         //$row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS2']);
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 }
```



## Web.php

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

## Level.blade

```
resources > views > 🌐 level.blade.php > ⚙ html
1  <!DOCTYPE html>
2  <html lang="en">
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>
```

← ⏪ ⓘ 127.0.0.1:8000/level

## 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 Facede 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')-&gt;get();</code>	<code>select * from m_kategori</code>
<code>DB::table('m_kategori')-&gt;where('kategori_id', 1)-&gt;get();</code>	<code>select * from m_kategori where kategori_id = 1;</code>
<code>DB::table('m_kategori')-&gt;select('kategori_kode')-&gt;where('kategori_id', 1)-&gt;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`

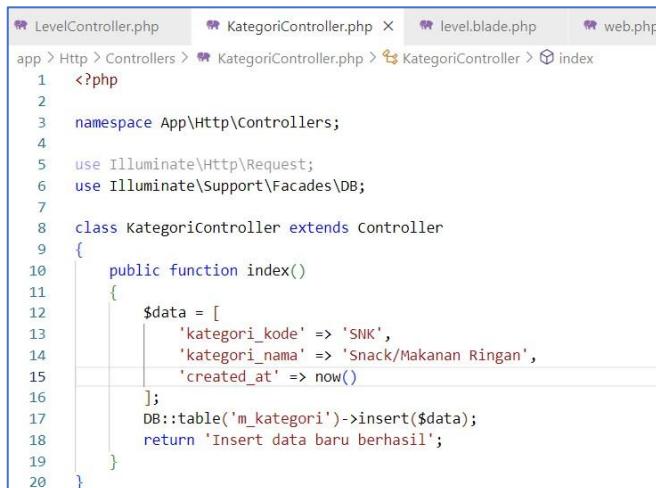
```
php artisan make:controller KategoriController
```

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

```
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']);
```



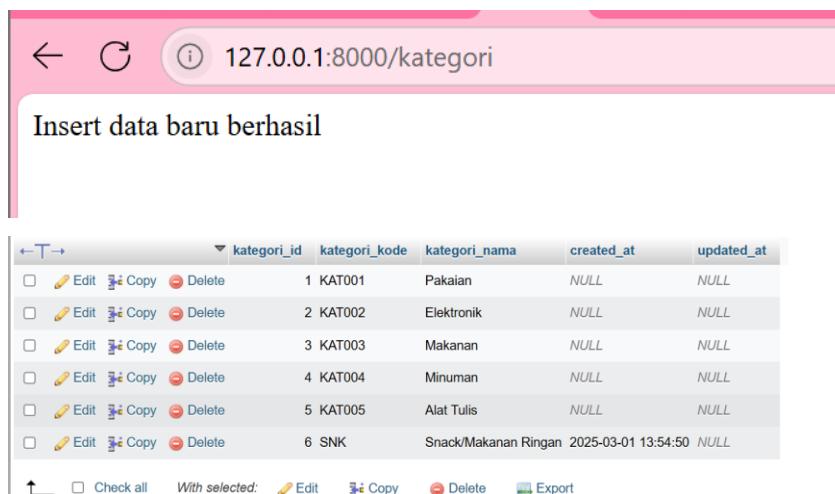
3. Selanjutnya, kita modifikasi file **KategoriController** untuk menambahkan 1 data ke table **m\_kategori**



```
LevelController.php KategoriController.php X level.blade.php web.php
app > Http > Controllers > KategoriController.php > 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](http://localhost/PWL_POS/public/kategori) dan amati apa yang terjadi pada table **m\_kategori** di database, *screenshot* perubahan yang ada pada table **m\_kategori**

#### Jawab



The screenshot shows a browser window with the URL [127.0.0.1:8000/kategori](http://127.0.0.1:8000/kategori). The page displays the message "Insert data baru berhasil". Below this, a table lists the data in the m\_kategori table:

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	1	KAT001	Pakaian	NULL	NULL
<input type="checkbox"/>	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	3	KAT003	Makanan	NULL	NULL
<input type="checkbox"/>	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	5	KAT005	Alat Tulis	NULL	NULL
<input type="checkbox"/>	6	SNK	Snack/Makanan Ringan	2025-03-01 13:54:50	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](http://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**

**Jawab:**

The screenshot shows a browser window with the URL [127.0.0.1:8000/kategori](http://127.0.0.1:8000/kategori). The page displays a success message: "Update data berhasil. Jumlah data yang diupdate: 1 baris". Below this, a table lists six categories with their details. The last row, which has "SNK" in the "kategori\_kode" column, is highlighted in grey, indicating it was updated.

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	1	KAT001	Pakaian	NULL	NULL
<input type="checkbox"/>	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	3	KAT003	Makanan	NULL	NULL
<input type="checkbox"/>	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	5	KAT005	Alat Tulis	NULL	NULL
<input type="checkbox"/>	6	SNK	Camilan	2025-03-01 13:54:50	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      // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
20      // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';

21      $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
22      return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row.' baris';
23  }
```

### Jawab

Delete data berhasil. Jumlah data yang diupdate: 1 baris

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	1	KAT001	Pakaian	NULL	NULL
<input type="checkbox"/>	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	3	KAT003	Makanan	NULL	NULL
<input type="checkbox"/>	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	5	KAT005	Alat Tulis	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      // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
20      // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';

21      // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
22      // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row.' baris';

23      $data = DB::table('m_kategori')->get();
24      return view('kategori', ['data' => $data]);
25  }
```



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.

### Jawab



The screenshot shows a browser window with the URL `127.0.0.1:8000/kategori`. The page title is "Data kategori Barang". Below the title is a table with the following data:

ID	Kode Kategori	Nama Kategori
1	KAT001	Pakaian
2	KAT002	Elektronik
3	KAT003	Makanan
4	KAT004	Minuman
5	KAT005	Alat Tulis

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>
```

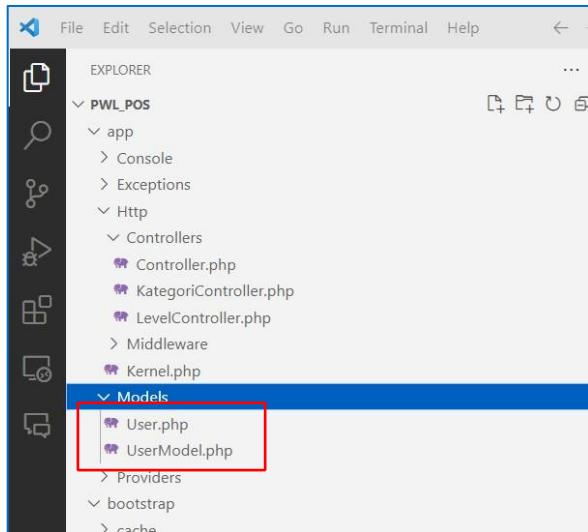
Untuk bisa melakukan operasi ***CRUD*** (*create, read/retrieve, update, delete*), kita harus membuat sebuah model sesuai dengan target tabel yang ingin digunakan. Jadi,  
**dalam 1 model, merepresentasikan 1 tabel database.**



## Praktikum 6 – Implementasi Eloquent ORM

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

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



2. 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`
3. 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
```



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']);
16
```

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 }
```

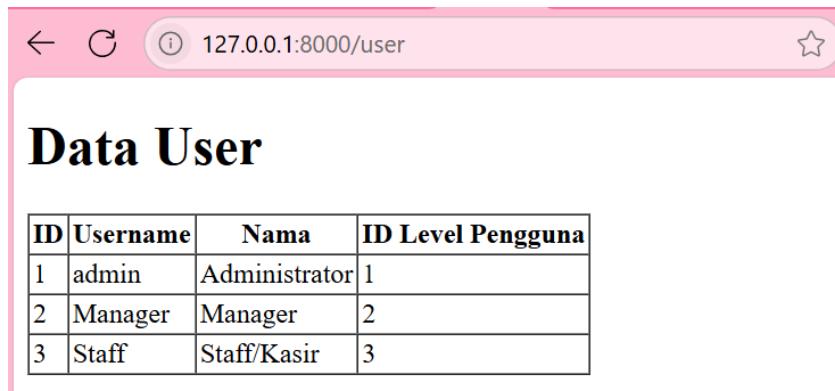
6. Kemudian kita buat view [user.blade.php](#)

```
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 laporan apa yang terjadi

**Jawab**



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 laporan 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

**Jawab**

The screenshot shows a web browser window with a pink header bar. The address bar displays '127.0.0.1:8000/user'. The main content area has a title 'Data User' and a table with the following data:

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

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

## G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari **APP\_KEY** pada *file setting .env* Laravel?

**Jawab:**

APP\_KEY adalah sebuah konfigurasi pada file '.env' di Laravel yang digunakan untuk menyimpan kunci enkripsi aplikasi. Kunci ini digunakan untuk menjaga keamanan data yang dienkripsi, seperti session cookies dan data lainnya

2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk **APP\_KEY**?

**Jawab:**

Untuk meng-*generate* nilai 'APP\_KEY', kita dapat menggunakan perintah artisan didalam terminal dengan menggunakan '**php artisan key:generate**'

3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi?



dan untuk apa saja file migrasi tersebut?

**Jawab:**

Secara default, Laravel memiliki dua file migrasi: ‘`create_users_table.php`’ dan ‘`create_password_resets_table.php`’. File ini digunakan untuk membuat table pengguna(users) dan reset password

4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/*output* dari fungsi tersebut?

**Jawab:**

Kode ‘`$table->timestaps();`’ secara otomatis menambahkan dua kolom, yaitu ‘`created_at`’ dan ‘`updated_at`’ yang digunakan untuk melacak waktu pembuatan dan pembaruan suatu record dalam tabel.

5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?

**Jawab:**

Fungsi ‘`$table->ide();`’ menghasilkan tipe data ‘`unsigned big integer`’ yang secara otomatis bertambah nilainya (auto-incrementing). Digunakan untuk menentukan kolom primary key dalam sebuah tabel

6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?

**Jawab:**

Perbedaan antara menggunakan ‘`$table->id();`’ dan ‘`$table->id('level_id');`’ pada hasil migrasi pada tabel ‘`m_level`’ menggunakan ‘`->id('level_id')`’ memberi nama kolom primary key secara eksplisit, yaitu ‘`level_id`’. Sedangkan ‘`->id();`’ secara implisit.

7. Pada migration, Fungsi `->unique()` digunakan untuk apa?

**Jawab:**

Fungsi ‘`->unique()`’ pada migration digunakan untuk menentukan nilai pada kolom tersebut harus unik, tidak boleh ada duplikat nilai antara baris-baris dalam tabel.

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')` ?

**Jawab:**



Kolom ‘level\_id’ pada tabel ‘m\_user’ menggunakan ‘\$table->unsignedBigInteger(‘level\_id’)’ diasumsikan sebagai **foreign key** yang merujuk ke primary key di tabel lain. Sedangkan pada tabel ‘m\_level’, ‘\$table->id(‘level\_id’)’ digunakan untuk mendefinisikan primary key dengan nama kolom ‘level\_id’.

9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?

**Jawab:**

Class ‘Hash’ digunakan untuk melakukan enkripsi data, seperti enkripsi password pengguna. Kode program ‘Hash::make(‘1234’);’ menghasilkan hasil enkripsi dari string ‘1234’ yang biasanya digunakan saat membuat dan menyimpan password baru pengguna ke dalam database.

10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?

**Jawab:**

Tanda tanya(?) pada query builder digunakan untuk menandai parameter yang akan dibind secara dinamis ke dalam query, memungkinkan pengguna parameter query dan melindungi terhadap serangan SQL Injection.

11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` ?

**Jawab:**

Penulisan kode berikut adalah bagian dari model Eloquent dalam framework laravel. Tujuan dari kode ini adalah untuk menghubungkan model tersebut dengan tabel tertentu dalam basis data dan menetapkan primary key dari tabel tersebut.

12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan

**Jawab:**

Menurut pendapat saya lebih mudah jika menggunakan Query Builder dikarenakan memiliki sintaks yang digunakan untuk mempermudah penggunaan query dan juga sintaks tersebut mudah dipahami disbanding dengan Eloquent Orm yang sintaksnya sulit untuk dipahami dan juga pada DB Façade penulisan sintaksnya masih manual



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI  
**POLITEKNIK NEGERI MALANG**  
**JURUSAN TEKNOLOGI INFORMASI**  
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141  
Telp. (0341) 404424 – 404425, Fax (0341) 404420  
<http://www.polinema.ac.id>

---

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