



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**



The screenshot shows the phpMyAdmin interface for a MySQL server named 'localhost'. In the center, there's a 'Databases' section with a 'Create database' button. A text input field contains 'PWL\_POS' and a dropdown menu shows 'utf8mb4\_unicode\_ci' selected. A 'Create' button is visible at the bottom right of the input area.

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**.

The screenshot shows the VSCode code editor with the '.env' file open. The file contains Laravel configuration variables. The 'APP\_NAME' is set to 'Laravel', 'APP\_ENV' to 'local', and 'APP\_URL' to 'http://localhost'. The 'DB\_CONNECTION' is set to 'mysql', 'DB\_HOST' to '127.0.0.1', 'DB\_PORT' to '3306', 'DB\_DATABASE' to 'PWL\_POS', 'DB\_USERNAME' to 'root', and 'DB\_PASSWORD' to an empty string. The file is saved in a folder named 'PWL\_POS' which also contains 'app', 'bootstrap', 'config', 'database', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', and other files like '.editorconfig' and '.gitattributes'.

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

The screenshot shows the VSCode code editor with the '.env' file open. The 'APP\_KEY' variable is highlighted with a red box, containing the value 'base64:KgPEif3b6D0mqm2FxJEy3lHh13EFasEJDuXXn9Af22Y=' (a random string). The 'DB\_CONNECTION', 'DB\_HOST', 'DB\_PORT', 'DB\_DATABASE', 'DB\_USERNAME', and 'DB\_PASSWORD' variables are also highlighted with a red box, containing their respective values: 'mysql', '127.0.0.1', '3306', 'PWL\_POS', 'root', and an empty string respectively. The file structure is identical to the previous screenshot, showing the 'PWL\_POS' folder with various Laravel files.

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



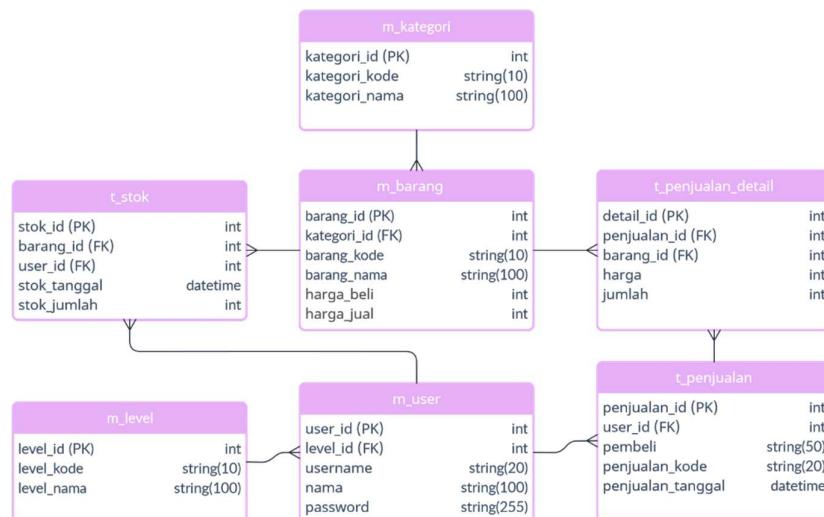
## B. MIGRATION

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 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	<a href="#">t_stok</a>	2
7	<a href="#">t_penjualan_detail</a>	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*

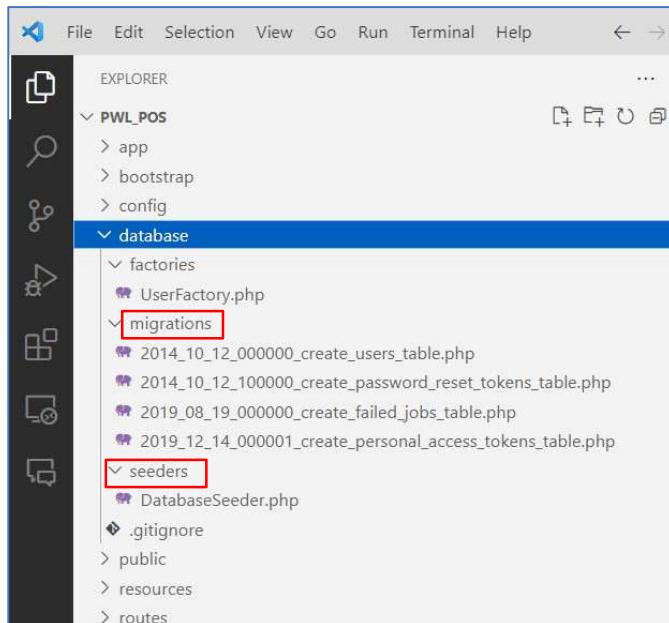
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- 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.

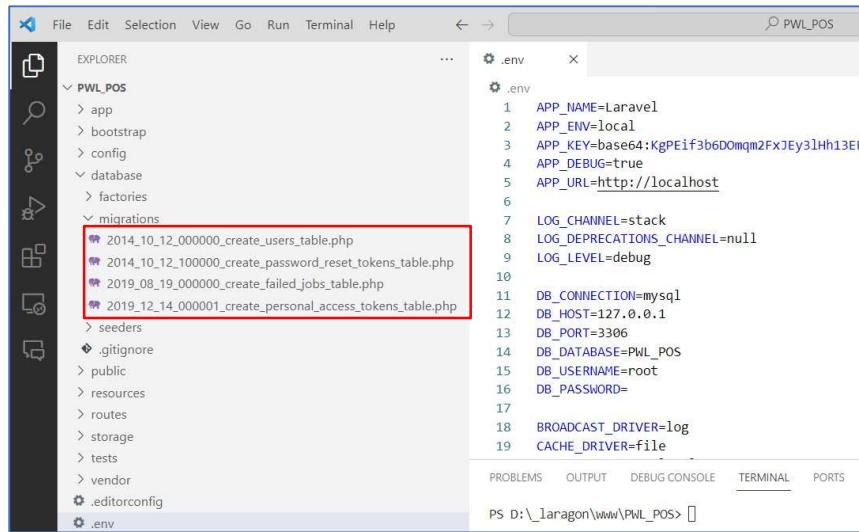
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder [PWL\\_POS/database](#)





## Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

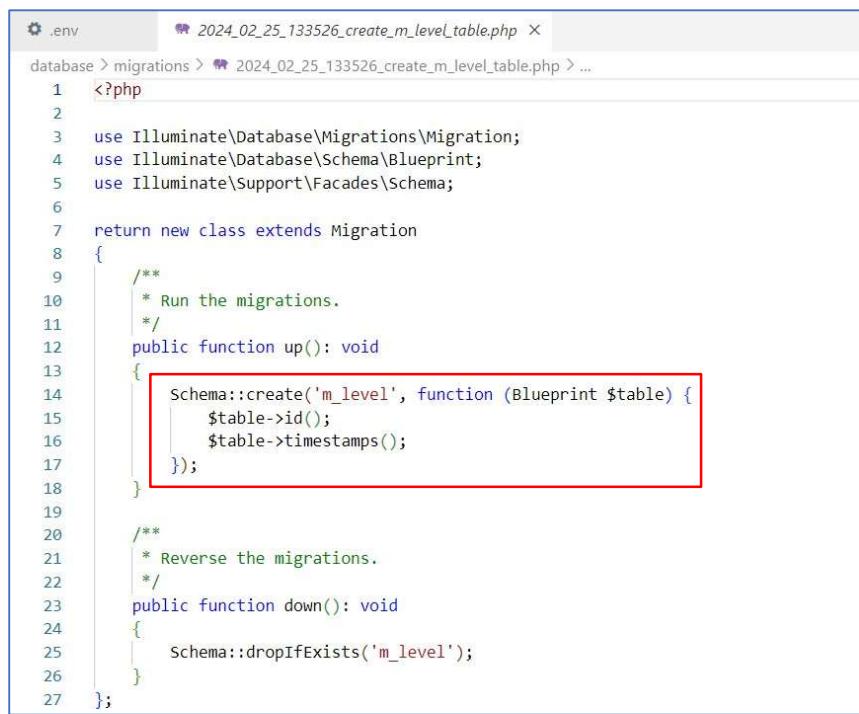
1. Buka *terminal* VSCode kalian, untuk yang di kotak merah adalah default dari laravel



```
File Edit Selection View Go Run Terminal Help ⌘ PWL_POS
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2FxJEy3lHh13EF
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=
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      * 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}
```

#### 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	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0
migrations	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	5
<b>m_level</b>	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0
password_reset_tokens	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0
personal_access_tokens	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0
users	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0

7. Ok, table sudah dibuat di database  
8. Buat table *database* dengan *migration* untuk table **m\_kategori** yang sama-sama tidak memiliki *foreign key*  
9. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*.

```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS> php artisan make:migration create_m_level_table --create=m_level
[INFO] Migration [D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS\database\migrations\2025_03_05_162017_create_m_level_table.php] created successfully.

PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS> php artisan migrate
[INFO] Preparing database.
Creating migration table ..... 55ms DONE
[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 43ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 8ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 30ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 34ms DONE
2025_03_05_162017_create_m_level_table ..... 24ms DONE

PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS> php artisan make:migration create_m_kategori_table --create=m_kategori
[INFO] Migration [D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS\database\migrations\2025_03_05_162714_create_m_kategori_table.php] created successfully.

PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS> php artisan migrate
[INFO] Running migrations.

2025_03_05_162714_create_m_kategori_table ..... 25ms DONE

PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Minggu3\PWL_POS>
```

## 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.

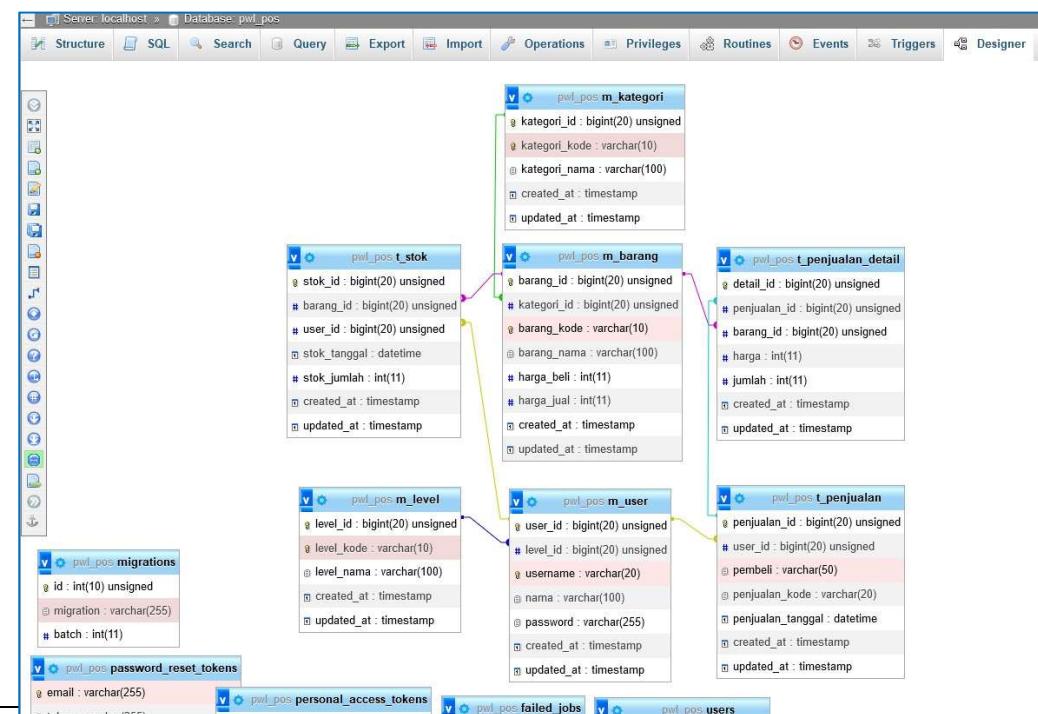
```
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Winggu3\PWL_POS> php artisan migrate
INFO  Running migrations.
2025_03_05_163623_create_m_user_table ..... 69ms DONE
PS D:\Documents\Polinema\STUDI KAMPUS\SEMESTER 4\WEB\PWL2025\Winggu3\PWL_POS> php artisan make:migration create_m_barang_table --table=m_barang
```

Table	Action	Rows	Type	Collation	Size
failed_jobs	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB
migrations	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	7	InnoDB	utf8mb4_unicode_ci	16.0 KiB
m_kategori	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB
m_level	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB
m_user	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB
password_reset_tokens	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB
personal_access_tokens	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB
users	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB
8 tables		Sum		7 InnoDB utf8mb4_0900_ai_ci	144.0 KiB

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

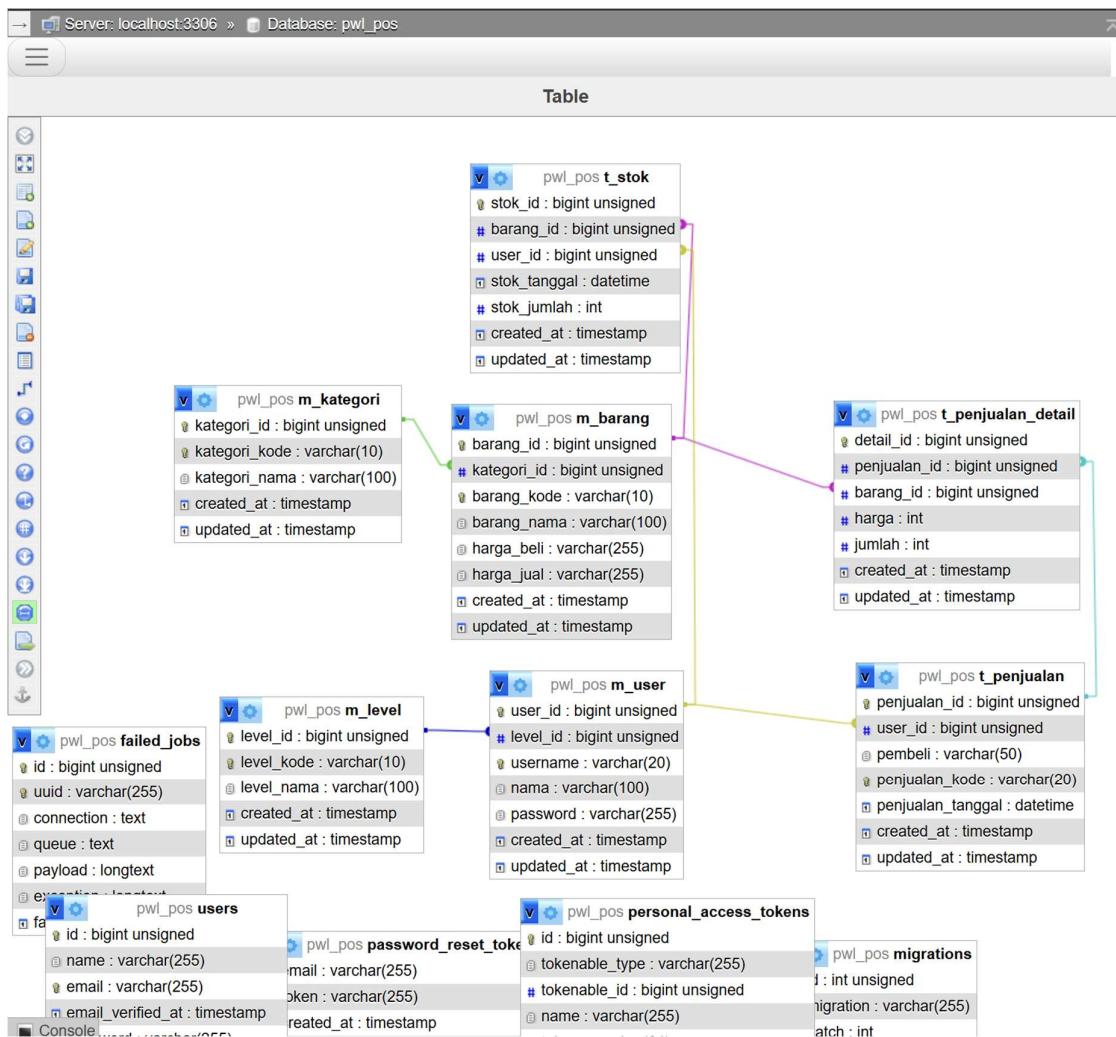
m_barang
t_penjualan
t_stok
t_penjualan_detail

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





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



Untuk hasil dari praktikum 2.2 ada pada database tersebut, sudah ada tabel , kolom, dan foreign key nya masing-masing



## 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/seeders**

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

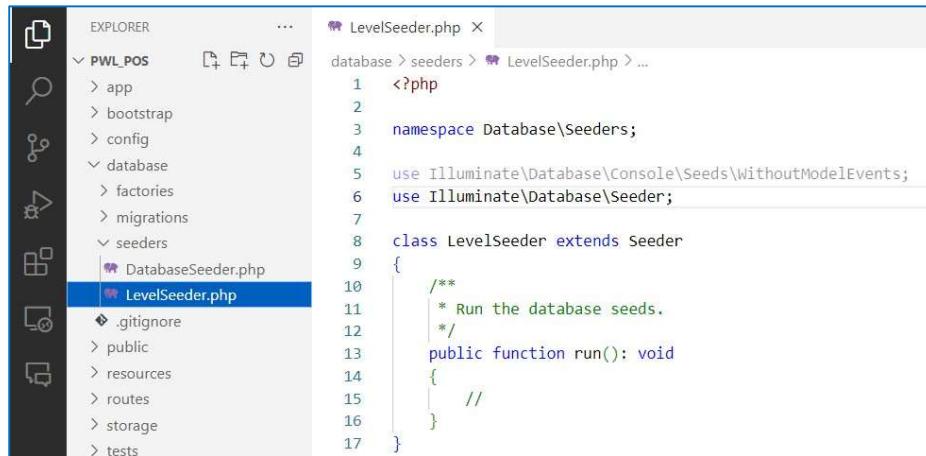
```
php artisan db:seed --class=<nama-class-seeder>
```

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

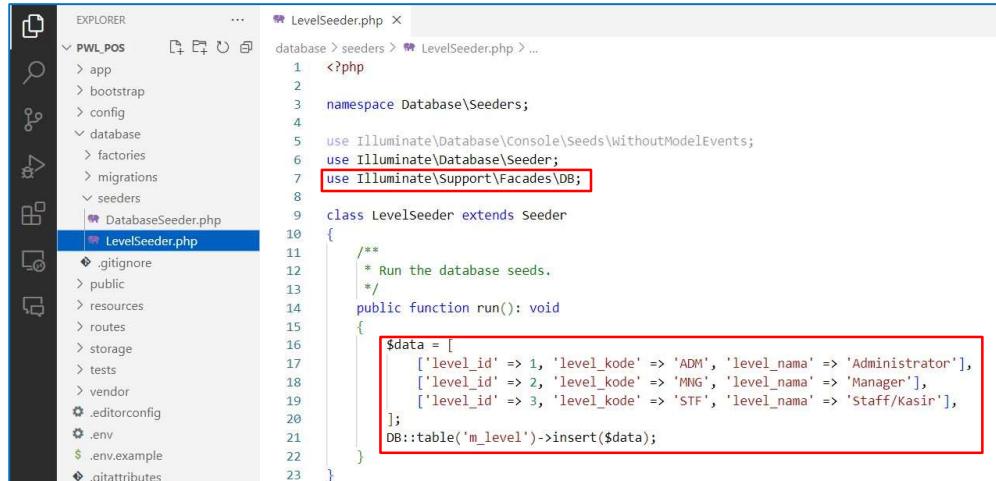
```
php artisan make:seeder LevelSeeder
```



```
LevelSeeder.php
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
8 class LevelSeeder extends Seeder
9 {
10
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         //
17     }
}
```



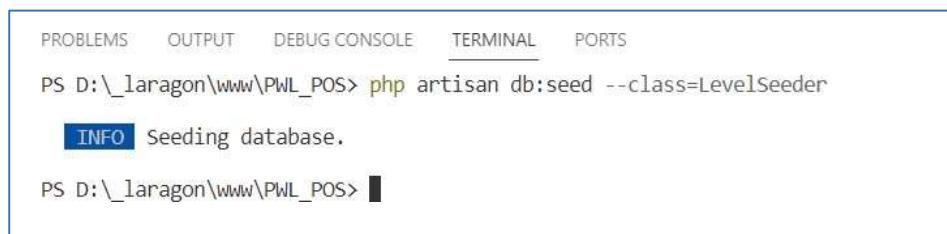
2. Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function `run()`



```
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"/>	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.LySwhY.4oAKU7FzwS6XV...
<input type="checkbox"/>		2	2	manager	Manager	\$2y\$12\$Ajfnz20/FdPTeUgghz31muEhlFaruLxkh5wvZ9NGRp...
<input type="checkbox"/>		3	3	staff	Staff/Kasir	\$2y\$12\$GI23TqGclW5pYeR0VL4o5OxPwb3Osk99VMy/BHnbJ9W...

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

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



*t\_kategori*

	<input type="button" value="←"/> <input type="button" value="→"/>	kategori_id	kategori.kode	kategori.nama	created_at	updated_at
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	KSM	Kendaraan Sepeda Motor	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	AMK	Alat Makan	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	PKS	Perkakas	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	ATK	Alat Tulis Kerja	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	5	TPW	Tupperware	NULL	NULL

*m\_barang*

	<input type="button" value="←"/> <input type="button" value="→"/>	kategori_id	kategori.kode	kategori.nama	created_at	updated_at
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	KSM	Kendaraan Sepeda Motor	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	AMK	Alat Makan	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	PKS	Perkakas	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	ATK	Alat Tulis Kerja	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	5	TPW	Tupperware	NULL	NULL

*t\_stok*

	<input type="button" value="←"/> <input type="button" value="→"/>	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	1	1	2025-03-05 17:59:11	20	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	2	1	2025-03-05 17:59:11	15	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	3	1	2025-03-05 17:59:11	50	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	4	2	2025-03-05 17:59:11	40	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	5	5	2	2025-03-05 17:59:11	100	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	6	6	3	2025-03-05 17:59:11	30	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	7	7	3	2025-03-05 17:59:11	25	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	8	8	1	2025-03-05 17:59:11	10	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	9	9	2	2025-03-05 17:59:11	60	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	10	10	3	2025-03-05 17:59:11	20	NULL	NULL

*t\_penjualan*



		penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/>	Edit  Copy  Delete	1	3	Pembeli1	PJ0001	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	2	3	Pembeli2	PJ0002	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	3	3	Pembeli3	PJ0003	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	4	3	Pembeli4	PJ0004	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	5	3	Pembeli5	PJ0005	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	6	3	Pembeli6	PJ0006	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	7	3	Pembeli7	PJ0007	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	8	3	Pembeli8	PJ0008	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	9	3	Pembeli9	PJ0009	2025-03-05 18:02:22	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	10	3	Pembeli10	PJ0010	2025-03-05 18:02:22	NULL	NULL

### *t\_penjualan\_detail*

		detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>	Edit  Copy  Delete	1	1	2	200000	5	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	2	1	5	30000	4	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	3	1	8	55000	1	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	4	2	6	70000	4	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	5	2	7	5000	1	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	6	2	10	45000	2	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	7	3	1	60000	3	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	8	3	4	7000	5	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	9	3	7	5000	1	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	10	4	2	200000	3	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	11	4	6	70000	2	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	12	4	8	55000	3	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	13	5	2	200000	3	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	14	5	3	10000	1	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	15	5	7	5000	2	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	16	6	2	200000	3	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	17	6	5	30000	1	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	18	6	6	70000	4	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	19	7	4	7000	4	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	20	7	6	70000	4	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	21	7	8	55000	5	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	22	8	3	10000	3	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	23	8	5	30000	4	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	24	8	7	5000	2	NULL	NULL
<input type="checkbox"/>	Edit  Copy  Delete	25	9	1	60000	4	NULL	NULL



## 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']);
```

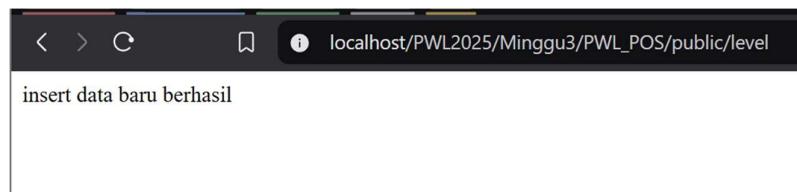
3. Selanjutnya, kita modifikasi file **LevelController** untuk menambahkan 1 data ke table **m\_level**

```
LevelController.php  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](localhost/PWL_POS/public/level) dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`

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



Jawaban: Akan terjadi penambahan data pada tabel `m_level`

	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	CUS	Pelanggan	2025-03-05 18:17:54	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](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`



< > C ⌂ ① localhost/PWL2025/Minggu3/PWL\_POS/public/level

Update data berhasil. Jumlah data yang diupdate: 1 baris

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 }
```

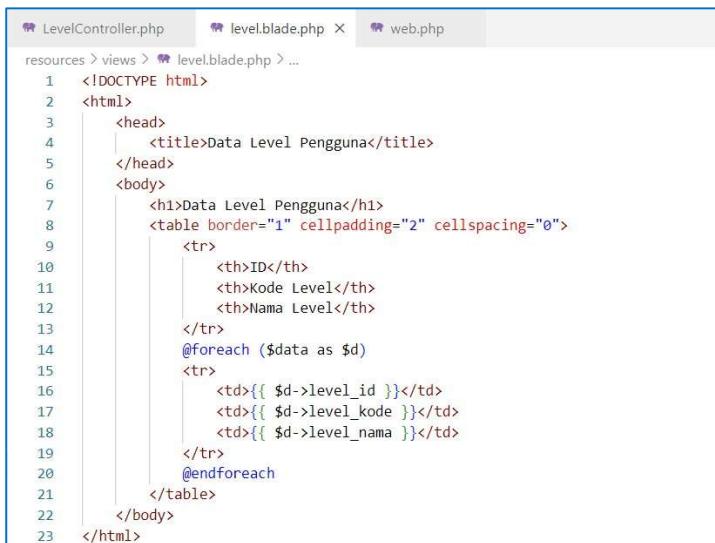
← ↑ →	▼	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit			1 ADM	Administrator	NULL
<input type="checkbox"/>	Edit			2 MNG	Manager	NULL
<input type="checkbox"/>	Edit			3 STF	Staff/Kasir	NULL



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

Menampilkan data level pengguna



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>

---

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

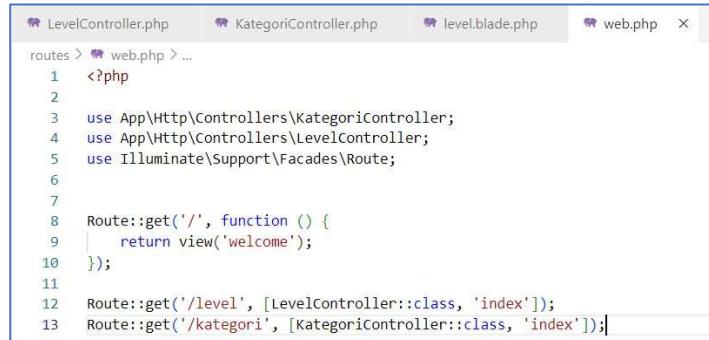
Method Query Builder	Query yang dihasilkan
<code>DB::table('m_kategori')-&gt;get();</code>	<code>select * from m_kategori</code>
<code>DB::table('m_kategori')-&gt;where('kategori_id', 1)-&gt;get();</code>	<code>select * from m_kategori where kategori_id = 1;</code>
<code>DB::table('m_kategori')-&gt;select('kategori_kode')-&gt;where('kategori_id',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

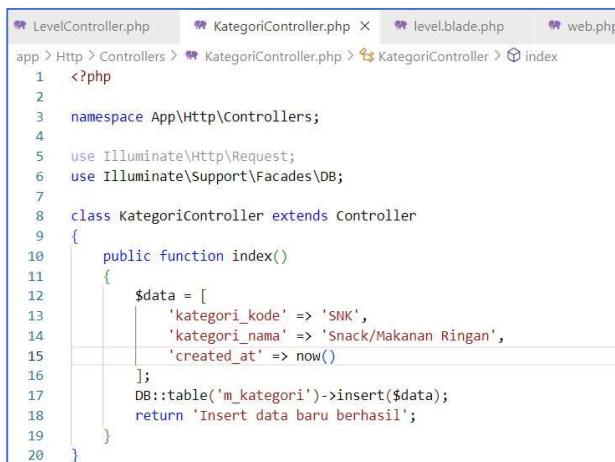
```
php artisan make:controller KategoriController
```

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



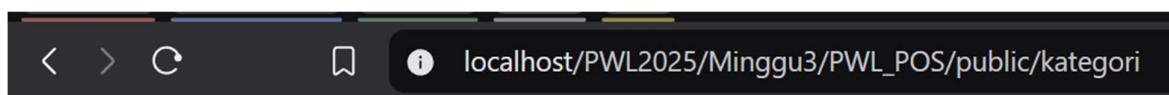
```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use Illuminate\Support\Facades\Route;
6
7
8  Route::get('/', function () {
9      |     return view('welcome');
10 });
11
12 Route::get('/level', [LevelController::class,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

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



```
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](localhost/PWL_POS/public/kategori) dan amati apa yang terjadi pada table `m_kategori` di database, screenshot perubahan yang ada pada table `m_kategori`



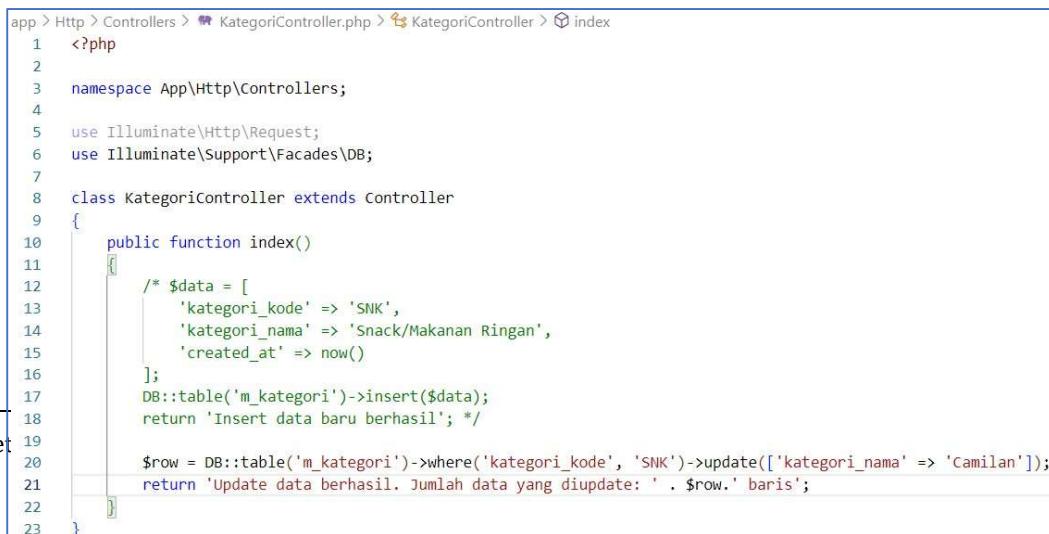
Insert data baru berhasil



	kategori_id	kategori.kode	kategori.nama	created_at	updated_at
<input type="checkbox"/>  Edit  Copy  Delete	1	KSM	Kendaraan Sepeda Motor	NULL	NULL
<input type="checkbox"/>  Edit  Copy  Delete	2	AMK	Alat Makan	NULL	NULL
<input type="checkbox"/>  Edit  Copy  Delete	3	PKS	Perkakas	NULL	NULL
<input type="checkbox"/>  Edit  Copy  Delete	4	ATK	Alat Tulis Kerja	NULL	NULL
<input type="checkbox"/>  Edit  Copy  Delete	5	TPW	Tupperware	NULL	NULL
<input type="checkbox"/>  Edit  Copy  Delete	6	SNK	Snack/Makanan Ringan	2025-03-05 18:36:14	NULL

#### Penambahan Data pada tabel kategori

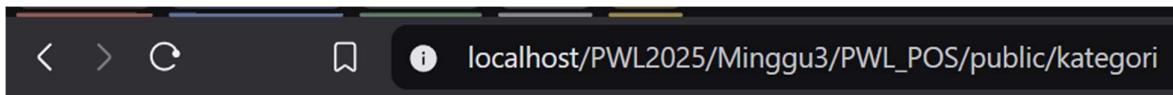
5. Selanjutnya, kita modifikasi lagi file `KategoriController` untuk meng-*update* data di table `m_kategori` seperti berikut



```
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         $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     }
24 }
```



6. Kita coba jalankan di browser dengan url [localhost/PWL\\_POS/public/kategori](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

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 }
```

< > ⏪ ⏩ 📌 ⓘ localhost/PWL2025/Minggu3/PWL\_POS/public/kategori

Delete data berhasil. Jumlah data yang dihapus: 1 baris

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 }
```

← ↑ →	▼	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>		Edit		Copy		Delete
<input type="checkbox"/>		Edit		Copy		Delete
<input type="checkbox"/>		Edit		Copy		Delete
<input type="checkbox"/>		Edit		Copy		Delete
<input type="checkbox"/>		Edit		Copy		Delete

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	KSM	Kendaraan Sepeda Motor
2	AMK	Alat Makan
3	PKS	Perkakas
4	ATK	Alat Tulis Kerja
5	TPW	Tupperware

Akan menampilkan data kategori barang

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

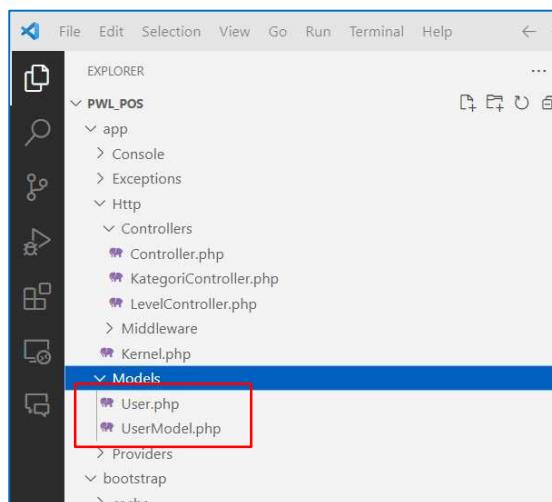
```
php artisan make:model <nama-model-CamelCase>
```

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']);
```

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

## Data User

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

Ada data user pada tabel m\_user

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 laporan apa yang terjadi

## Data User

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

12. Jika sudah, laporan 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?

Jawaban: APP\_KEY digunakan untuk enkripsi data, termasuk session dan password hashing.

2. Pada **Praktikum 1**, bagaimana kita men-generate nilai untuk `APP_KEY`?

Jawaban: 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?

Jawaban: create\_users\_table.php (tabel users), create\_password\_resets\_table.php (reset password), create\_failed\_jobs\_table.php (log job yang gagal)

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

Jawaban: \$table->timestamps(); akan membuat dua kolom: created\_at & updated\_at untuk mencatat waktu perubahan data.

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

Jawaban: \$table->id(); menghasilkan big integer (unsignedBigInteger) dengan auto-increment.

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

Jawaban: \$table->id(); membuat kolom id secara default, sedangkan \$table->id('level\_id'); membuat kolom level\_id sebagai primary key.

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

Jawaban: ->unique() digunakan untuk memastikan nilai dalam kolom tidak ada yang duplikat.

8. Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom `level_id` pada tabel `m_user` menggunakan `$table->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$table->id('level_id')` ?



Jawaban: m\_level menggunakan \$table->id('level\_id'); karena primary key, sedangkan m\_user menggunakan \$table->unsignedBigInteger('level\_id'); sebagai foreign key.

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

Jawaban: Class Hash digunakan untuk mengenkripsi password. `Hash::make('1234');` akan mengubah 1234 menjadi hash terenkripsi.

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

Jawaban: digunakan sebagai placeholder untuk parameter prepared statement

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

Jawaban: protected \$table = 'm\_user'; menentukan nama tabel di model. Sedangkan protected \$primaryKey = 'user\_id'; menentukan primary key yang digunakan.

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

Jawaban: Eloquent ORM lebih mudah karena lebih ringkas dibanding Query Builder atau DB Façade.

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