



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**.

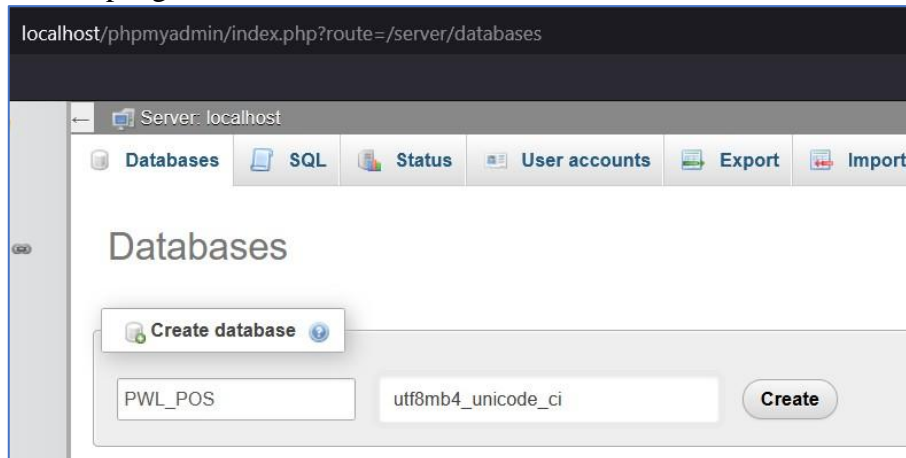
Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. PENGATURAN DATABASE

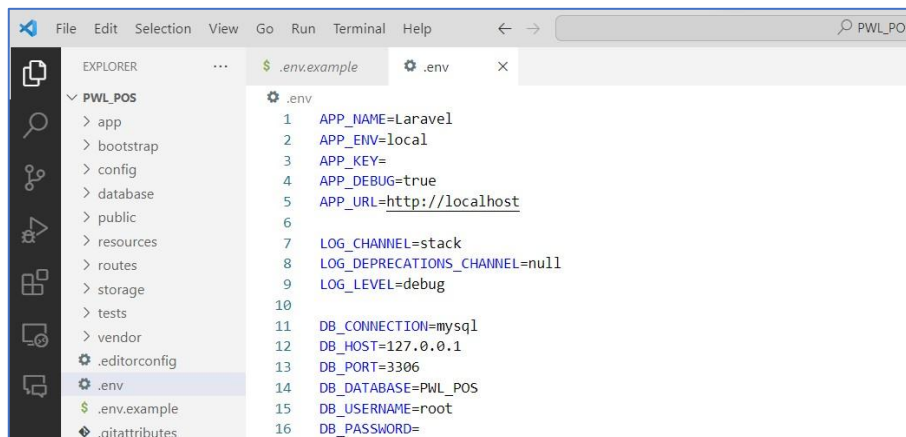
Database atau basis data menjadi komponen penting dalam membangun sistem. Hal ini dikarenakan database menjadi tempat untuk menyimpan data-data transaksi yang ada pada sistem. Koneksi ke database perlu kita atur agar sesuai dengan database yang kita gunakan.



Praktikum¹ - pengaturan database:



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



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

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



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2FxJey3lHh13EFaseJDuxXn9Af22Y=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=PWL_POS
15 DB_USERNAME=root
16 DB_PASSWORD=
```

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:n9t5BLsSwNogD1lp3e+gEyoQlGX1j61I00lKKnNCt40=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=pwl_pos
15 DB_USERNAME=root
16 DB_PASSWORD=
```

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

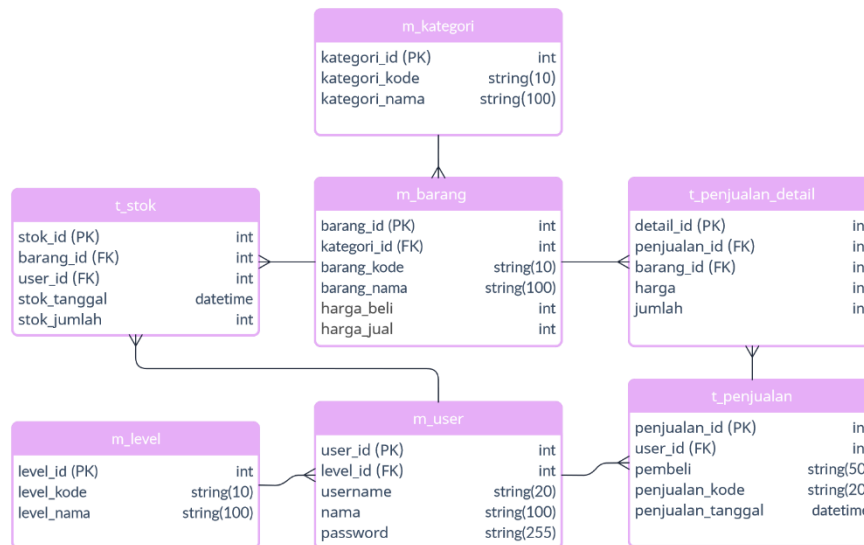
B. MIGRATION

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

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

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

Studi Kasus PWL.pdf



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

TIPS MIGRATION

Buatlah file migration untuk table yang tidak memiliki relasi (table yang tidak ada *foreign key*) dulu, dan dilanjutkan dengan membuat file migrasi yang memiliki relasi yang sedikit, dan 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	<code>m_level</code>	0
2	<code>m_kategori</code>	0
3	<code>m_user</code>	1
4	<code>m_barang</code>	1
5	<code>t_penjualan</code>	1
6	<code>t_stok</code>	2
7	<code>t_penjualan_detail</code>	2

INFO

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



Pembuatan file migrasi bisa menggunakan 2 cara, yaitu

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

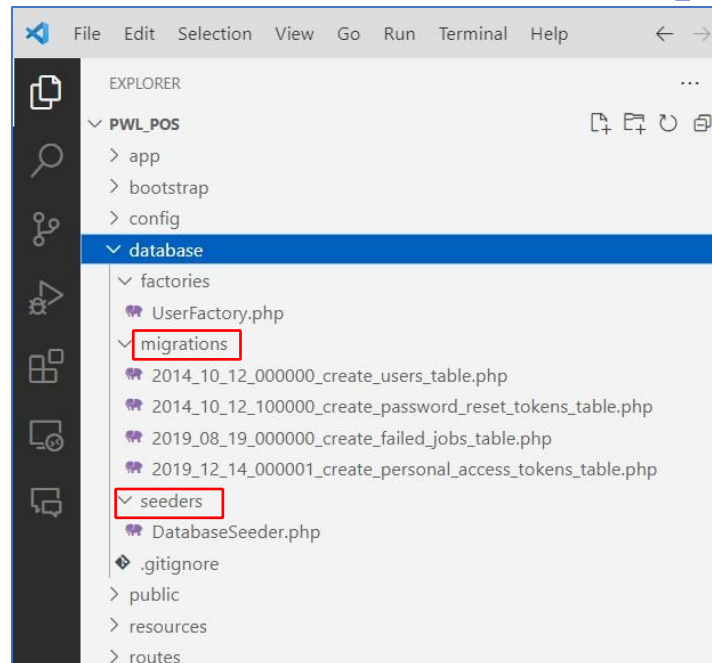
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

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

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

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

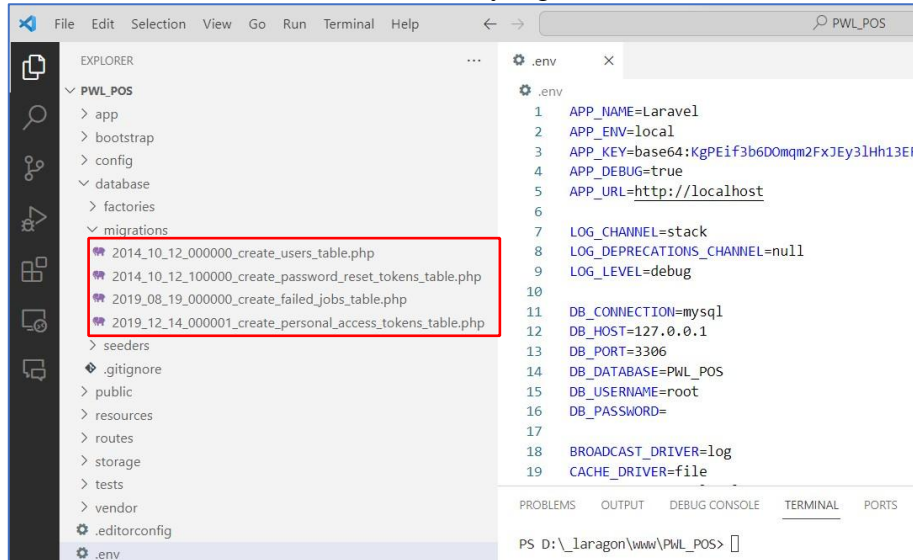
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder **PWL_POS/database**





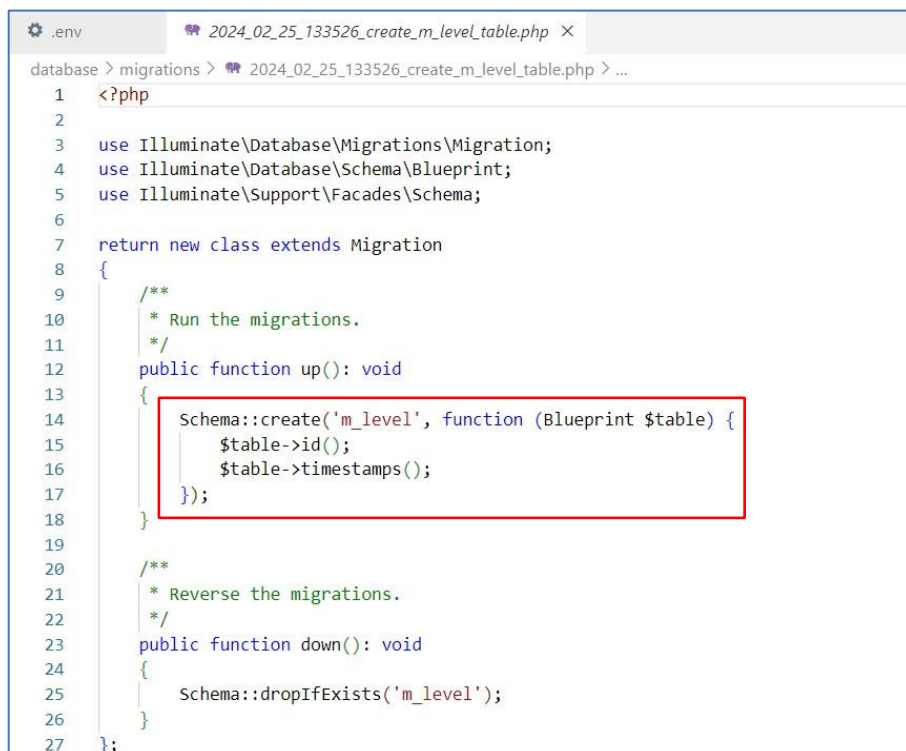
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



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

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





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 };
```

```
C:\laragon\www\PWL_POS>php artisan make:migration create_m_level_table -
--create=m_level
```

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

INFO

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

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

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



```
php artisan migrate
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

INFO Preparing database.

Creating migration table 12ms DONE

INFO Running migrations.

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

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

INFO Preparing database.

Creating migration table 485ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table 220ms DONE
2014_10_12_100000_create_password_reset_tokens_table 95ms DONE
2019_08_19_000000_create_failed_jobs_table 174ms DONE
2019_12_14_000001_create_personal_access_tokens_table 161ms DONE
2025_03_04_022608_create_m_level_table 55ms DONE

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

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

Hasil :

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

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



```
C:\laragon\www\PWL_POS>php artisan make:migration create_m_kategori_table --create=m_level
```

```
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create('m_kategori', function (Blueprint $table) {
15               $table->id('kategori_id');
16               $table->string('kategori_kode', 10)->unique();
17               $table->string('kategori_nama', 100);
18               $table->timestamps();
19           });
20       }
21
22       /**
23        * Reverse the migrations.
24        */
25       public function down(): void
26       {
27           Schema::dropIfExists('m_kategori');
28       }
29   };
30
```

```
INFO Running migrations.
2025_03_04_023739_create_m_kategori_table ..... 91ms DONE
```

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
<input type="checkbox"/> migrations	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
<input type="checkbox"/> m_kategori	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
<input type="checkbox"/> m_level	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
<input type="checkbox"/> password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
<input type="checkbox"/> personal_access_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
<input type="checkbox"/> users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 K18	-
7 tables	Sum	6	InnoDB	utf8mb4_0900_ai_ci	112.0 K18	0 B

☐ Check all With selected:

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

². Buka *terminal* VSCode kalian, dan buat file migrasi untuk table **m_user**

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



Praktikum 2.³ - Pembuatan file migrasi dengan relasi

```
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.

INFO Running migrations.

2025_03_04_033803_create_m_user_table 120ms **DONE**

Table	Action	Rows	Type	Collation	Size	Overhead
failed_jobs	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
migrations	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	9	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_kategori	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_level	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_user	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
password_reset_tokens	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
personal_access_tokens	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
users	✱ Browse 🔍 Structure 🔍 Search 🔍 Insert 🔍 Empty 🔍 Drop 🔍	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
8 tables	Sum	9	InnoDB	utf8mb4_0900_ai_ci	144.0 KiB	0 B

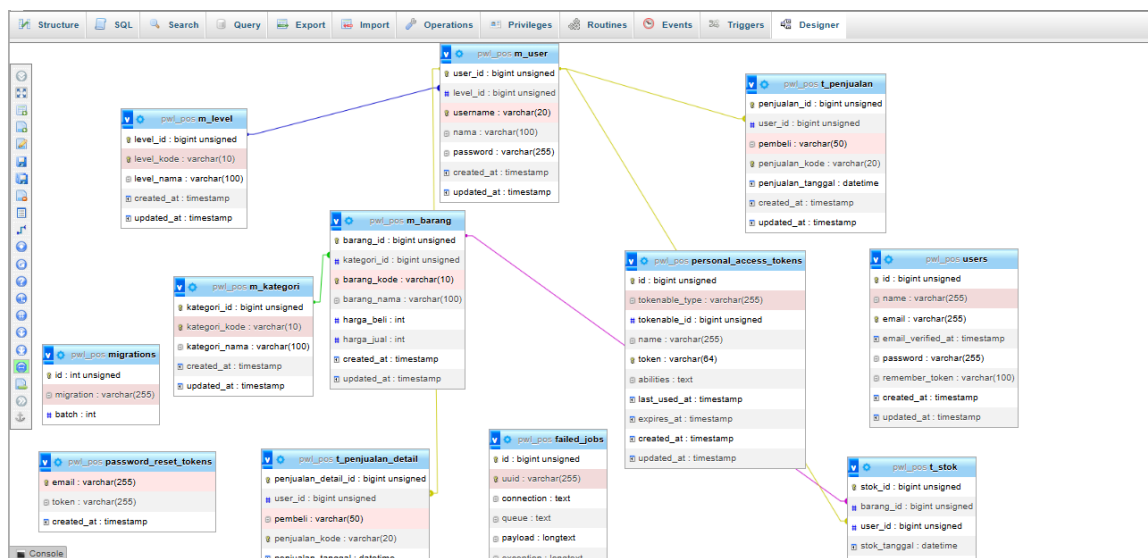
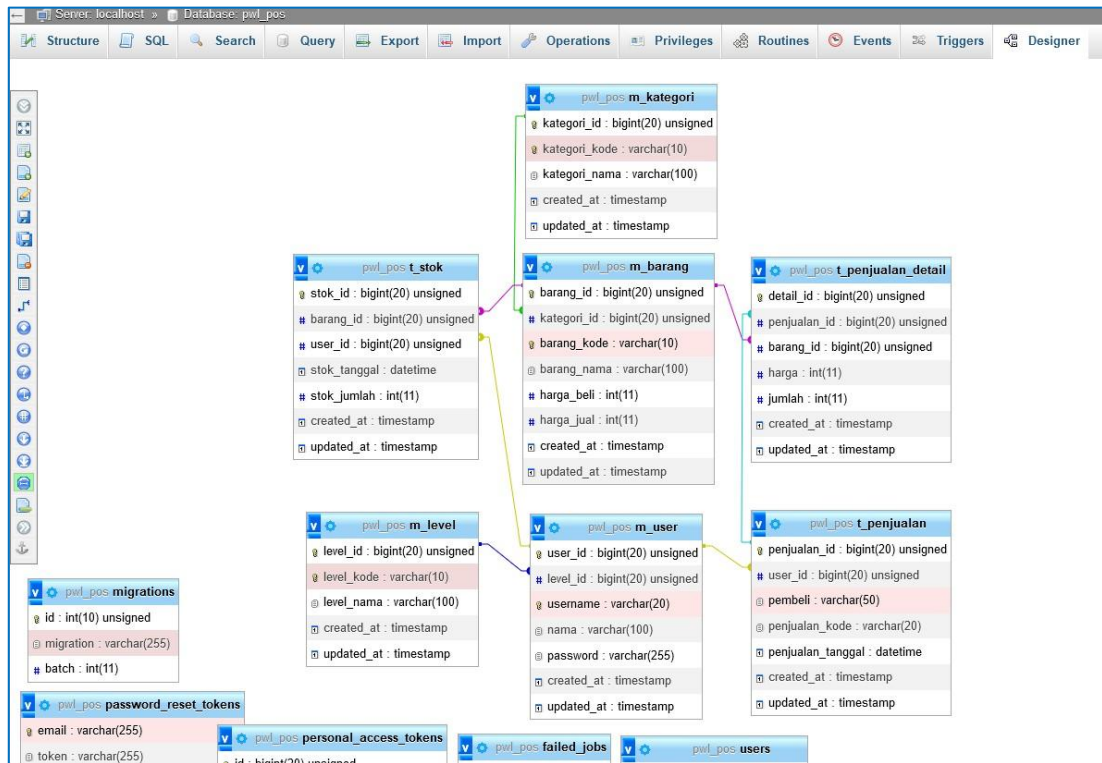
🔍 ☐ Check all

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*.



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/seeder**s

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

```
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()`

```
<?php
namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
            ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
            ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
        ];
        DB::table('m_level')->insert($data);
    }
}
```

Hasil:

```
<?php
namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
            ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
            ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
        ];
        DB::table('m_level')->insert($data);
    }
}
```

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

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=LevelSeeder
```

```
INFO Seeding database.
```

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



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

Hasil :

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

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

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

```
C:\laragon\www\PWL_POS>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 }
```



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

8. Perhatikan hasil seeder pada table **m_user**

	user_id	level_id	username	nama	password
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6fXV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhIFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...

☐ Check all With selected: ☐ Edit ☐ Copy ☐ Delete ☐ Export

8. Jalankan perintah untuk mengeksekusi class **UserSeeder**

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$24QNHXmTFqjFAQDUHgS02OCFvnlIF3JgmowMj4m2/Rk...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$117ftOyUjvsfLkrGhPoLmeYd8nsmDggQOF7IP8o4KYj...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$svWIRCrjdGJCIm0n3W.JO8AWby6FCUVrguTETB4IXQ...	NULL	NULL

9. Ok, data seeder berhasil di masukkan ke database.

10. Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut

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

- Kategori

```
C:\laragon\www\PWL_POS>php artisan make:seeder KategoriSeeder
```

```
database > seeders > KategoriSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriSeeder extends Seeder
9  {
10     public function run(): void
11     {
12         $data = [
13             ['kategori_id' => 1, 'kategori_kode' => 'FB', 'kategori_nama' => 'Food & Beverage'],
14             ['kategori_id' => 2, 'kategori_kode' => 'BH', 'kategori_nama' => 'Beauty & Health'],
15             ['kategori_id' => 3, 'kategori_kode' => 'HC', 'kategori_nama' => 'Home Care'],
16             ['kategori_id' => 4, 'kategori_kode' => 'BK', 'kategori_nama' => 'Baby & Kids'],
17             ['kategori_id' => 5, 'kategori_kode' => 'EL', 'kategori_nama' => 'Electronics'],
18         ];
19
20         DB::table('m_kategori')->insert($data);
21     }
22 }
```



```
C:\laragon\www\PWL_POS>php artisan db:seed --class=KategoriSeeder
```

INFO Seeding database.

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	FB	Food & Beverage	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	BH	Beauty & Health	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	HC	Home Care	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	BK	Baby & Kids	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	EL	Electronics	NULL	NULL

- Barang

```
C:\laragon\www\PWL_POS>php artisan make:seeder BarangSeeder
```

```
database > seeders > BarangSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7
8  class BarangSeeder extends Seeder
9  {
10     public function run(): void
11     {
12         $data = [
13             ['barang_id' => 1, 'kategori_id' => 1, 'barang_kode' => 'B001', 'barang_nama' => 'Air Mineral', 'harga beli'
14             ['barang_id' => 2, 'kategori_id' => 1, 'barang_kode' => 'B002', 'barang_nama' => 'Roti Tawar', 'harga beli'
15             ['barang_id' => 3, 'kategori_id' => 2, 'barang_kode' => 'B003', 'barang_nama' => 'Sabun Mandi', 'harga beli'
16             ['barang_id' => 4, 'kategori_id' => 2, 'barang_kode' => 'B004', 'barang_nama' => 'Shampoo', 'harga beli' =>
17             ['barang_id' => 5, 'kategori_id' => 3, 'barang_kode' => 'B005', 'barang_nama' => 'Pembersih Lantai', 'harga
18             ['barang_id' => 6, 'kategori_id' => 3, 'barang_kode' => 'B006', 'barang_nama' => 'Tisu', 'harga beli' => 800
19             ['barang_id' => 7, 'kategori_id' => 4, 'barang_kode' => 'B007', 'barang_nama' => 'Popok Bayi', 'harga beli'
20             ['barang_id' => 8, 'kategori_id' => 4, 'barang_kode' => 'B008', 'barang_nama' => 'Susu Formula', 'harga beli'
21             ['barang_id' => 9, 'kategori_id' => 5, 'barang_kode' => 'B009', 'barang_nama' => 'Lampu LED', 'harga beli' =
22             ['barang_id' => 10, 'kategori_id' => 5, 'barang_kode' => 'B010', 'barang_nama' => 'Charger HP', 'harga beli'
23         ];
24         DB::table('m_barang')->insert($data);
25     }
26 }
27
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=BarangSeeder
```

INFO Seeding database.

		barang_id	kategori_id	barang_kode	barang_nama	harga beli	harga jual	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	B001	Air Mineral	2000	3000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	1	B002	Roti Tawar	10000	12000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	2	B003	Sabun Mandi	5000	7000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	2	B004	Shampoo	15000	18000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	3	B005	Pembersih Lantai	10000	13000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	3	B006	Tisu	8000	10000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	4	B007	Popok Bayi	50000	55000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	4	B008	Susu Formula	60000	65000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	5	B009	Lampu LED	25000	30000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	5	B010	Charger HP	35000	40000	NULL	NULL

↑ ☐ Check all With selected: Edit Copy Delete Export



- Stok

```
C:\laragon\www\PWL_POS>php artisan make:seeder StokSeeder
```

```
database > seeders > StokSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7  use Carbon\Carbon;
8
9  class StokSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data = [];
14         for ($i = 1; $i <= 10; $i++) {
15             $data[] = [
16                 'barang_id' => $i,
17                 'user_id' => 1, // Admin yang melakukan input stok
18                 'stok_tanggal' => Carbon::now(),
19                 'stok_jumlah' => rand(10, 100),
20             ];
21         }
22         DB::table('t_stok')->insert($data);
23     }
24 }
25
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=StokSeeder
```

INFO Seeding database.

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	1	1	1	2025-03-04 04:55:26	93	NULL	NULL
<input type="checkbox"/>	2	2	2	2025-03-04 04:55:26	73	NULL	NULL
<input type="checkbox"/>	3	3	3	2025-03-04 04:55:26	79	NULL	NULL
<input type="checkbox"/>	4	4	4	2025-03-04 04:55:26	52	NULL	NULL
<input type="checkbox"/>	5	5	5	2025-03-04 04:55:26	78	NULL	NULL
<input type="checkbox"/>	6	6	6	2025-03-04 04:55:26	55	NULL	NULL
<input type="checkbox"/>	7	7	7	2025-03-04 04:55:26	30	NULL	NULL
<input type="checkbox"/>	8	8	8	2025-03-04 04:55:26	25	NULL	NULL
<input type="checkbox"/>	9	9	9	2025-03-04 04:55:26	33	NULL	NULL
<input type="checkbox"/>	10	10	10	2025-03-04 04:55:26	76	NULL	NULL

- Penjualan

```
C:\laragon\www\PWL_POS>php artisan make:seeder PenjualanSeeder
```

```
database > seeders > PenjualanSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7  use Carbon\Carbon;
8
9  class PenjualanSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data = [];
14         for ($i = 1; $i <= 10; $i++) {
15             $data[] = [
16                 'user_id' => rand(1, 3), // Kasir yang berbeda
17                 'pembeli' => 'Customer ' . $i,
18                 'penjualan_kode' => 'TRX00' . $i,
19                 'penjualan_tanggal' => Carbon::now()->subDays(rand(1, 30)),
20             ];
21         }
22         DB::table('t_penjualan')->insert($data);
23     }
24 }
25
```



```
C:\laragon\www\PWL_POS>php artisan db:seed --class=PenjualanSeeder
```

INFO Seeding database.

	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/>	1	3	Customer 1	TRX001	2025-02-13 05:05:13	NULL	NULL
<input type="checkbox"/>	2	3	Customer 2	TRX002	2025-02-05 05:05:13	NULL	NULL
<input type="checkbox"/>	3	1	Customer 3	TRX003	2025-02-02 05:05:13	NULL	NULL
<input type="checkbox"/>	4	3	Customer 4	TRX004	2025-03-03 05:05:13	NULL	NULL
<input type="checkbox"/>	5	1	Customer 5	TRX005	2025-02-20 05:05:13	NULL	NULL
<input type="checkbox"/>	6	1	Customer 6	TRX006	2025-02-06 05:05:13	NULL	NULL
<input type="checkbox"/>	7	2	Customer 7	TRX007	2025-02-03 05:05:13	NULL	NULL
<input type="checkbox"/>	8	2	Customer 8	TRX008	2025-02-11 05:05:13	NULL	NULL
<input type="checkbox"/>	9	1	Customer 9	TRX009	2025-02-16 05:05:13	NULL	NULL
<input type="checkbox"/>	10	3	Customer 10	TRX0010	2025-02-15 05:05:13	NULL	NULL

• Penjualan_detail

```
C:\laragon\www\PWL_POS>php artisan make:seeder PenjualanDetailSeeder
```

```
database > seeders > PenjualanDetailSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7
8  class PenjualanDetailSeeder extends Seeder
9  {
10     public function run(): void
11     {
12         $data = [];
13         for ($i = 1; $i <= 10; $i++) { // 10 transaksi
14             for ($j = 1; $j <= 3; $j++) { // 3 barang per transaksi
15                 $barang_id = rand(1, 10);
16                 $barang = DB::table('m_barang')->where('barang_id', $barang_id)->first();
17
18                 $data[] = [
19                     'penjualan_id' => $i,
20                     'barang_id' => $barang_id,
21                     'harga' => $barang->harga_jual,
22                     'jumlah' => rand(1, 5),
23                 ];
24             }
25         }
26
27         DB::table('t_penjualan_detail')->insert($data);
28     }
29 }
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=PenjualanDetailSeeder
```

INFO Seeding database.

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>	1	1	3	7000	5	NULL	NULL
<input type="checkbox"/>	2	1	9	30000	2	NULL	NULL
<input type="checkbox"/>	3	1	10	40000	5	NULL	NULL
<input type="checkbox"/>	4	2	9	30000	4	NULL	NULL
<input type="checkbox"/>	5	2	1	3000	3	NULL	NULL
<input type="checkbox"/>	6	2	2	12000	5	NULL	NULL
<input type="checkbox"/>	7	3	7	55000	4	NULL	NULL
<input type="checkbox"/>	8	3	1	3000	3	NULL	NULL
<input type="checkbox"/>	9	3	8	65000	3	NULL	NULL
<input type="checkbox"/>	10	4	8	65000	4	NULL	NULL
<input type="checkbox"/>	11	4	10	40000	2	NULL	NULL
<input type="checkbox"/>	12	4	2	12000	2	NULL	NULL
<input type="checkbox"/>	13	5	6	10000	5	NULL	NULL
<input type="checkbox"/>	14	5	10	40000	4	NULL	NULL
<input type="checkbox"/>	15	5	9	30000	1	NULL	NULL
<input type="checkbox"/>	16	6	4	18000	5	NULL	NULL
<input type="checkbox"/>	17	6	5	13000	1	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

```
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']);
```

mengembalikan (*return*) data hasil *query*. Contoh

b. `DB::insert()`

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



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

c.

`DB::update()`

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

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

d. `DB::delete()`

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

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

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

Praktikum 4 – Implementasi DB Facade

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

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

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

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

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



m_level

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

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

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

Hasil saat dijalankan di local host dan perubahan database:

localhost/PWL_2025/Week3/Jobsheet3/public/level

Pretty print ☐

```
{"message": "Data berhasil ditambahkan"}
```

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	4	CUS	Pelanggan	2025-03-04 11:39:05	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`

← → ↺ ⓘ localhost/PWL_2025/Week3/Jobsheet3/public/level

Update data berhasil. Jumlah data yang diupdate: 1 baris

← T →		level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	CUS	Customer	2025-03-04 11:39:05	NULL

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

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



localhost/PWL_2025/Week3/Jobsheet3/public/level

Delete data berhasil. Jumlah data yang dihapus: 1 baris

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

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

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

9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil `view('level')`, maka kita buat file view pada VSCode di

`PWL_POS/resources/view/level.blade.php`

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

10. Silahkan dicoba pada browser dan amati apa yang terjadi



← → ↺ localhost/PWL_2025/Week3/Jobsheet3/public/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');
```

- b. 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
DB::table('m_kategori')->get();	select * from m_kategori
DB::table('m_kategori') ->where('kategori_id', 1)->get();	select * from m_kategori where kategori_id = 1;
DB::table('m_kategori') ->select('kategori_kode') ->where('kategori_id', 1)->get();	select kategori_kode from m_kategori where kategori_id = 1;



Praktikum 5 – Implementasi *Query Builder*

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

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

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

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

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

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

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

```
← → ↻ ⓘ localhost/PWL_2025/Week3/Jobsheet3/public/kategori
Insert data baru berhasil
```



		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	FB	Food & Beverage	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	BH	Beauty & Health	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	HC	Home Care	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	BK	Baby & Kids	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	EL	Electronics	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2025-03-04 12:15:06	NULL

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

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

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

localhost/PWL_2025/Week3/Jobsheet3/public/kategori

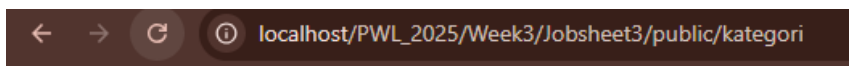
Update data berhasil. Jumlah data yang diupdate: 1 baris

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	FB	Food & Beverage	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	BH	Beauty & Health	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	HC	Home Care	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	BK	Baby & Kids	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	EL	Electronics	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Camilan	2025-03-04 12:15:06	NULL



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

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



Delete data berhasil. Jumlah data yang dihapus: 1 baris

			kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	FB	Food & Beverage	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	BH	Beauty & Health	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	HC	Home Care	NULL
<input type="checkbox"/>	Edit	Copy	Delete	4	BK	Baby & Kids	NULL
<input type="checkbox"/>	Edit	Copy	Delete	5	EL	Electronics	NULL

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

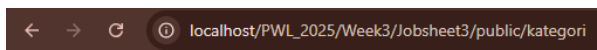
```
10 public function index()
11 {
12     /* $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17     DB::table('m_kategori')->insert($data);
18     return 'Insert data baru berhasil'; */
19
20     // $row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

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



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

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



Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	FB	Food & Beverage
2	BH	Beauty & Health
3	HC	Home Care
4	BK	Baby & Kids
5	EL	Electronics

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



F. ELOQUENT ORM

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

INFO

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

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

Perintah untuk membuat model adalah sebagai berikut

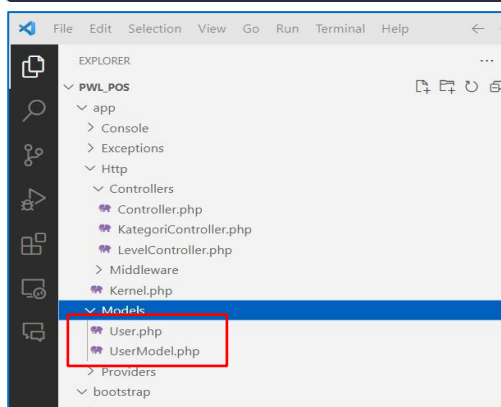
```
php artisan make:model <nama-model-CamelCase>
```

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

Praktikum 6 – Implementasi Eloquent ORM

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

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





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

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

- 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
```

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

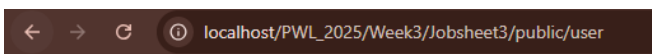
```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7
8  class UserController extends Controller
9  {
10     public function index()
11     {
12         // coba akses model UserModel
13  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

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



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

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



Data User

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

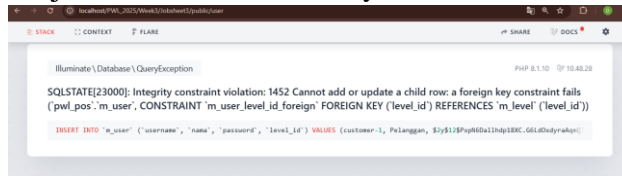
8. Setelah itu, kita modifikasi lagi file `UserController`

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



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

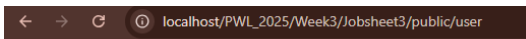
- Terjadi error, berikut hasilnya:



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

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

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



Data User

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

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?
 - Pada Laravel, `APP_KEY` merupakan kunci enkripsi utama yang digunakan untuk berbagai keperluan keamanan, terutama:
 - Mengenkripsi dan Mendekripsi Data
 - Keamanan Session dan Token CSRF
 - Mendukung Hashing Password yang Lebih Aman
2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk `APP_KEY`?
 - Untuk men-*generate* nilai `APP_KEY` di Laravel, Anda bisa menggunakan perintah Artisan berikut di terminal atau command prompt:
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?
 - Secara **default**, Laravel memiliki **4 file migrasi** yang terdapat di folder:
 - 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
4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/output dari fungsi tersebut?
 - Kode `$table->timestamps();` secara otomatis akan menambahkan dua kolom berikut ke tabel yang dibuat:
 - `created_at` → Untuk menyimpan tanggal & waktu saat data pertama kali dibuat.
 - `updated_at` → Untuk menyimpan tanggal & waktu saat data terakhir diperbarui.
 - Dengan adanya kedua kolom ini, Laravel dapat melacak kapan suatu data dibuat dan diupdate secara otomatis.
5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
 - Fungsi ini secara otomatis **membuat primary key** dengan tipe data



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

perbedaan	<code>\$table->id();</code>	<code>\$table->id('level_id');</code>
Nama Kolom	id (default Laravel)	level_id (custom)
Tipe	Data BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY	BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY
Penggunaan di Model	<code>\$model->id</code>	<code>\$model->level_id</code>

7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
- Fungsi `unique()` ini digunakan untuk membuat kolom dengan aturan unik, yang berarti nilai dalam kolom tersebut tidak boleh ada yang sama (duplikat).
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')` ?
- Menggunakan `$table->id('level_id')` karena `level_id` adalah Primary Key di `m_level`, sehingga perlu dibuat sebagai BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY agar setiap data memiliki ID unik yang bertambah otomatis.
 - Menggunakan `$table->unsignedBigInteger('level_id')` karena `level_id` di `m_user` bukan Primary Key, tetapi Foreign Key yang merujuk ke `m_level.level_id`. Oleh karena itu, tipe datanya harus cocok dengan primary key yang dirujuk (BIGINT UNSIGNED), tetapi tidak auto-increment.
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?
- Class `Hash` di Laravel digunakan untuk mengubah (hashing) password atau data sensitif lainnya agar lebih aman saat disimpan di database
 - Maksud dari `Hash::make('1234')` adalah untuk mengubah password menjadi bentuk yang lebih aman (hashing) sebelum disimpan di database.



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

- Tanda tanya (?) digunakan sebagai placeholder untuk parameter dalam query agar lebih aman dan menghindari SQL Injection.

11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table =`

`'m_user';` dan `protected $primaryKey = 'user_id';` ?

- `protected $table = 'm_user';`
Tujuan: Menentukan nama tabel yang digunakan oleh model.
- `protected $primaryKey = 'user_id';`
Tujuan: Menentukan kolom yang digunakan sebagai Primary Key.

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

- Menurut saya lebih mudah dengan Query Builder karena:
 - Lebih mudah digunakan dibandingkan SQL mentah.
 - Bisa digunakan tanpa harus membuat model.
 - Lebih fleksibel daripada Eloquent ORM.

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