



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 'Databases' section of the phpMyAdmin interface. A 'Create database' button is visible, with the database name 'PWL_POS' and character set 'utf8mb4_unicode_ci' selected. A 'Create' button is at the bottom right.

The screenshot shows the 'Structure' tab of the 'crud_db' table in the 'pwl_pos' database. It displays a 'Create new table' form with 'Table name' set to 'crud_db' and 'Number of columns' set to 4. A 'Create' button is at the bottom right.

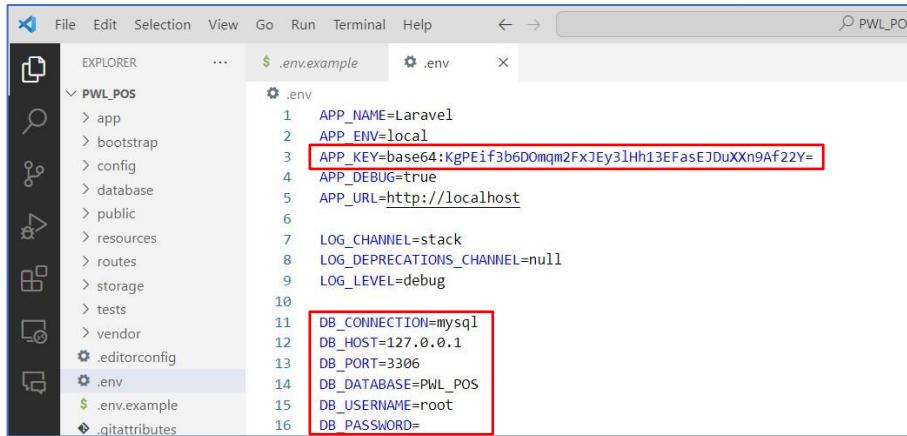
2. Buka aplikasi VSCode dan buka folder project **PWL_POS** yang sudah kita buat
3. Copy file **.env.example** menjadi **.env**
4. Buka file **.env**, dan pastikan konfigurasi **APP_KEY** bernilai. Jika belum bernilai silahkan kalian generate menggunakan **php artisan**.

The screenshot shows the VSCode code editor with the '.env' file open. The file contains the following Laravel configuration:

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=
APP_DEBUG=true
APP_URL=http://localhost
LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=PWL_POS
DB_USERNAME=root
DB_PASSWORD=
```

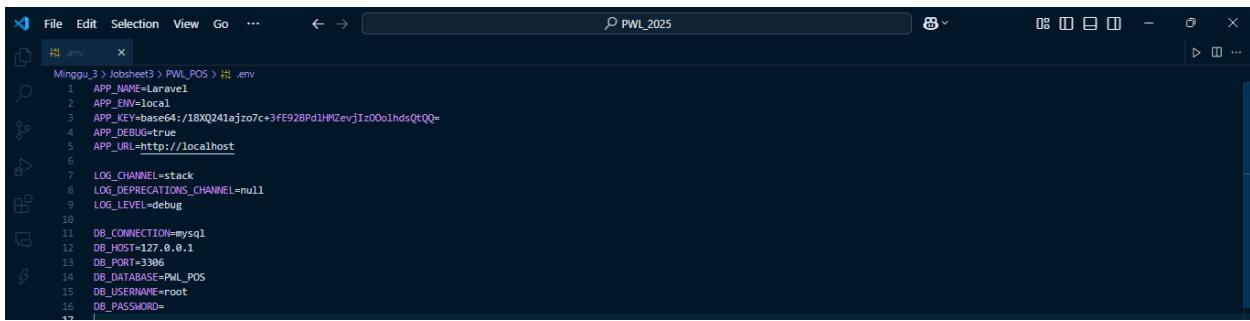


5. Edit file `.env` dan sesuaikan dengan database yang telah dibuat



```
File Edit Selection View Go Run Terminal Help ⏴ PWL_POS
EXPLORER ... $ .env.example .env ×
PWL_POS
> app
> bootstrap
> config
> database
> public
> resources
> routes
> storage
> tests
> vendor
> .editorconfig
> .env
> .env.example
$ .gitattributes
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2Fx3Ey3lHh13EFasEJDuXXn9Af22Y=
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. Laporan hasil Praktikum-1 ini dan *commit* perubahan pada *git*.



```
File Edit Selection View Go ⏴ PWL_2025
.env ×
Minggu_3 > Jobsheet3 > PWL_POS > .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:/18XQ241ajzo7c+3fE92BPd1HMZevjzOo01hsQtQQ=
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=
```



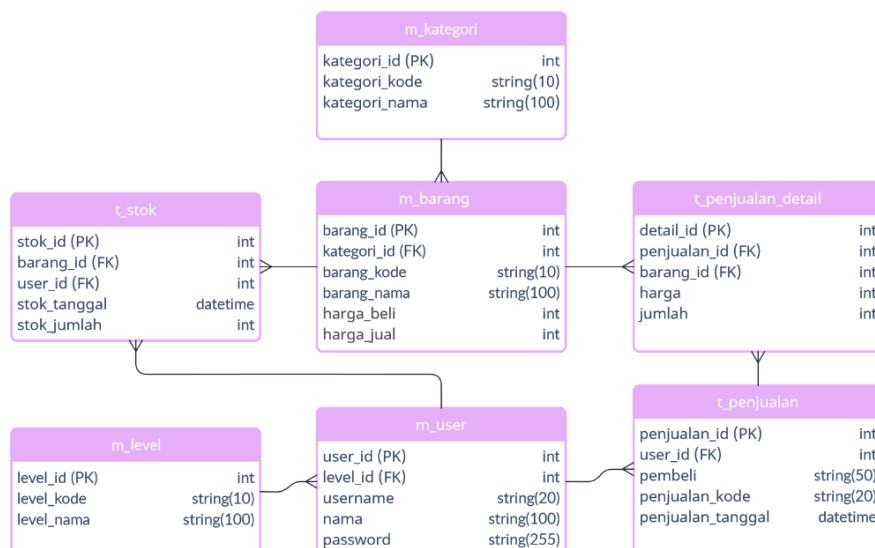
B. MIGRATION

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	m_level	0
2	m_kategori	0
3	m_user	1
4	m_barang	1
5	t_penjualan	1
6	t_stok	2
7	t_penjualan_detail	2

INFO

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

Pembuatan file migrasi bisa menggunakan 2 cara, yaitu

- Menggunakan [artisan](#) untuk membuat *file migration*

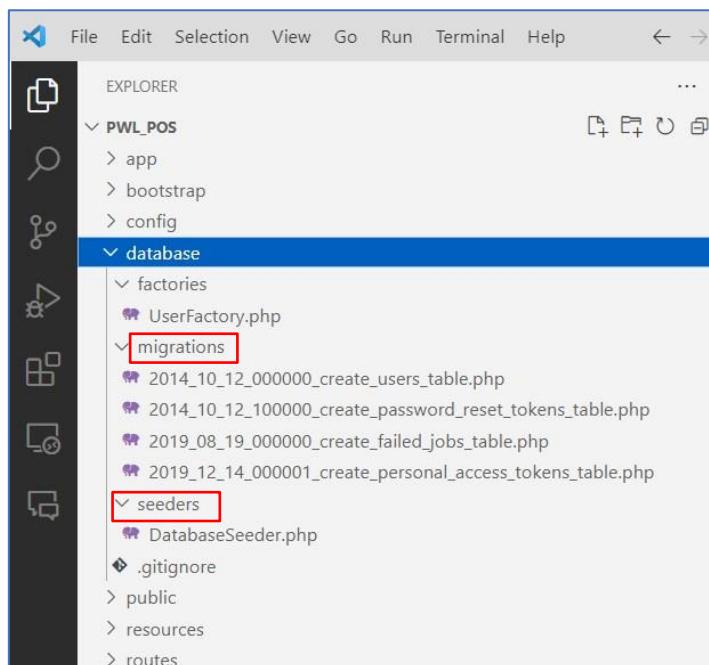
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- Menggunakan [artisan](#) untuk membuat *file model + file migration*

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

Perintah [-m](#) di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

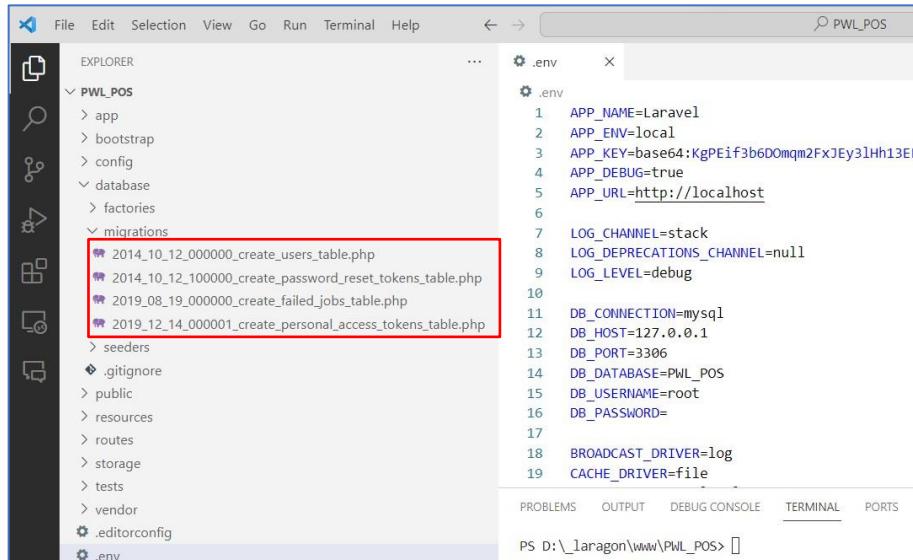
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder [PWL_POS/database](#)





Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

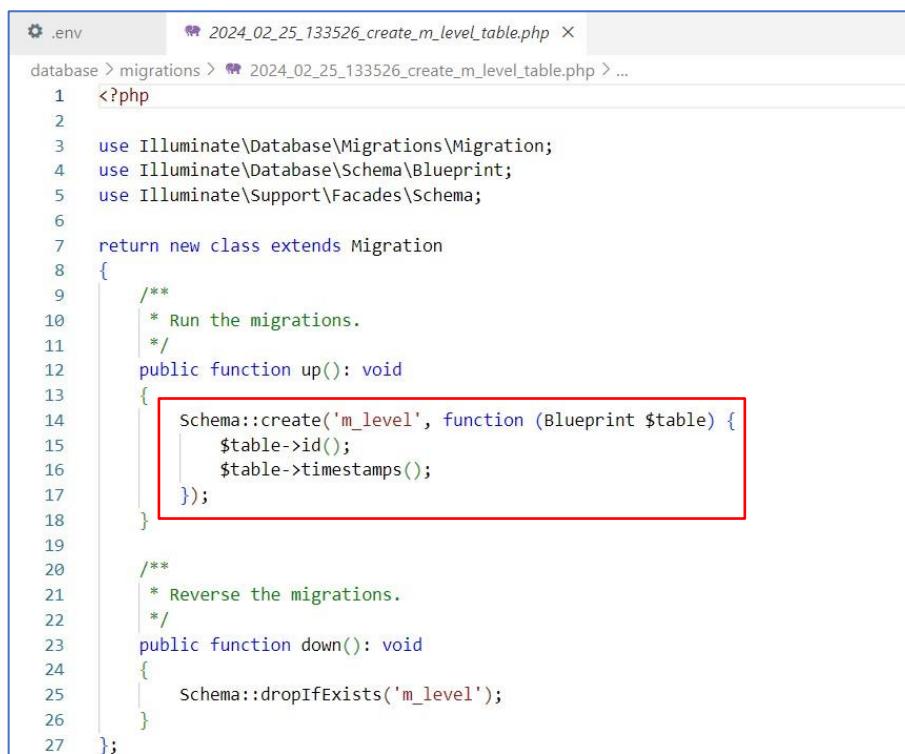
1. Buka *terminal* VSCode kalian, untuk yang di kotak merah adalah default dari laravel



```
File Edit Selection View Go Run Terminal Help
EXPLORER .env
PWL_POS
  app
  bootstrap
  config
  database
    factories
    migrations
      2014_10_12_000000_create_users_table.php
      2014_10_12_100000_create_password_reset_tokens_table.php
      2019_08_19_000000_create_failed_jobs_table.php
      2019_12_14_000001_create_personal_access_tokens_table.php
    seeders
    .gitignore
  public
  resources
  routes
  storage
  tests
  vendor
  .editorconfig
  .env
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64: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

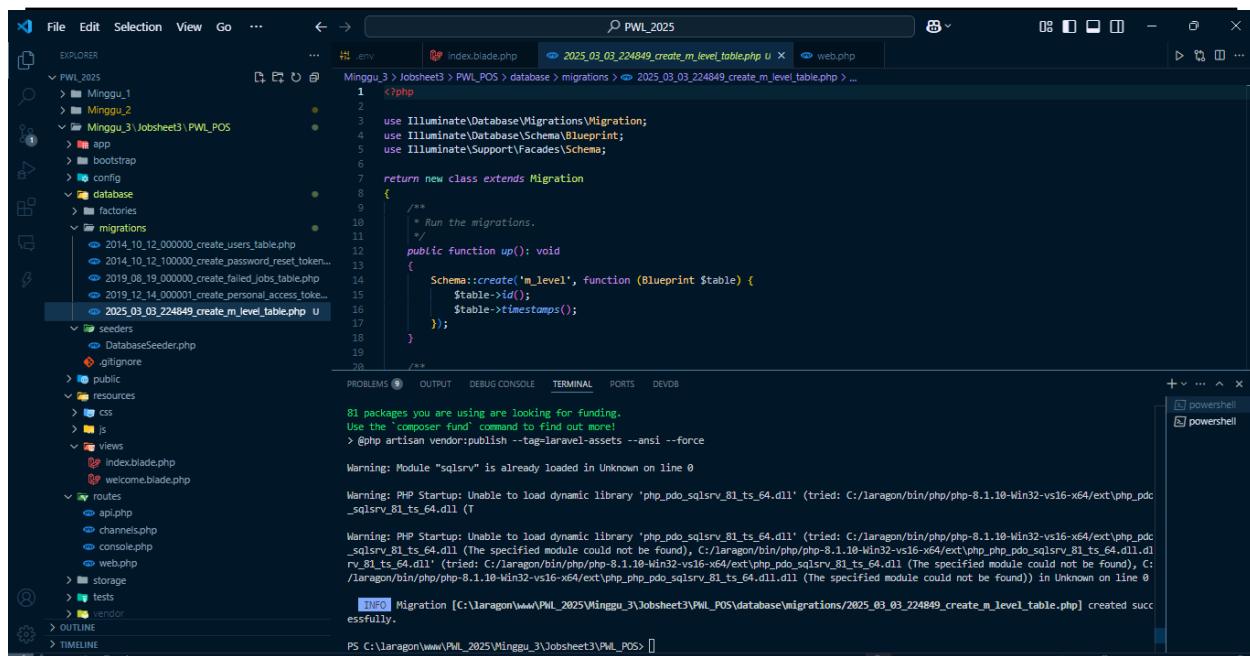
```
php artisan make:migration create_m_level_table --create=m_level
```



```
.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};
```



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>



The screenshot shows a code editor interface with a dark theme. On the left is a file explorer sidebar showing a project structure for a Laravel application named 'PWL_2025'. The 'migrations' folder contains several migration files, including one highlighted: '2025_03_03_224849_create_m_level_table.php'. The main editor area displays the PHP code for this migration:

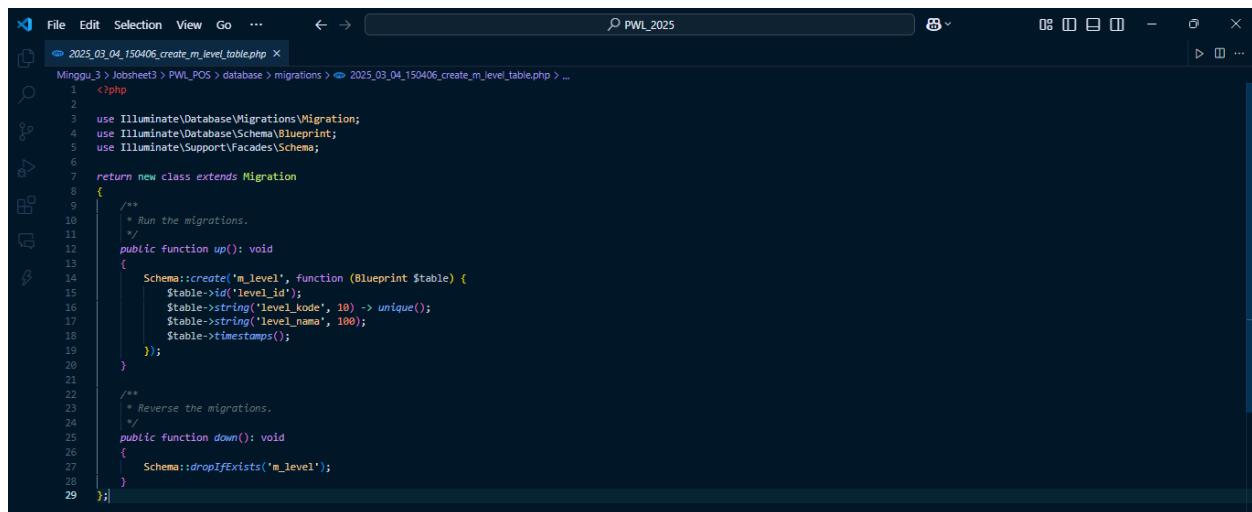
```
1 <?php
2
3     use Illuminate\Database\Migrations\Migration;
4     use Illuminate\Database\Schema\Blueprint;
5     use Illuminate\Support\Facades\Schema;
6
7     return new class extends Migration
8     {
9
10         /**
11          * Run the migrations.
12          */
13         public function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

The code editor includes tabs for 'index.blade.php', '2025_03_03_224849_create_m_level_table.php', and 'web.php'. Below the code editor is a 'PROBLEMS' panel showing several PHP startup warnings related to dynamic library loading. At the bottom, a terminal window shows the command 'composer fund' and the output of a migration creation command.



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  };
```



```
File Edit Selection View Go ... ← → PWL_2025 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

```
2025_03_04_150406_create_m_level_table.php
Minggu_3 > Jobsheet3 > PWL_POS > database > migrations > 2025_03_04_150406_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('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>
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVOB
[INFO] Preparing database.
Creating migration table ..... 169ms DONE
[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 265ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 36ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 115ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 31ms DONE
2025_03_03_224849_create_m_level_table ..... 80ms DONE

PS C:\laragon\www\PWL_2025\Winggu_3\Jobsheet3\PWL_POS>
```



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

The screenshot shows the phpMyAdmin interface for the 'pwl_pos' database. On the left, there's a sidebar with various database schemas like 'crud_db', 'information_schema', 'laravel', 'mysql', 'performance_schema', 'pwl_pos', and 'sys'. The main area displays a table of tables with columns for 'Table', 'Action', 'Rows', 'Type', 'Collation', and 'Size'. The 'm_level' table is highlighted with a red box. The table data is as follows:

Table	Action	Rows	Type	Collation	Size
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 Kib
migrations	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16.0 Kib
m_level	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 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

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



The screenshot shows the phpMyAdmin interface connected to the MySQL server at localhost:3306. The current database selected is 'pwl_pos'. The left sidebar lists various databases and their tables. The main area displays the table structure for 'pwl_pos' with the following details:

Table	Action	Rows	Type	Collation	Size
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 Kib
migrations	Browse Structure Search Insert Empty Drop	6	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
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
7 tables	Sum	6	InnoDB	utf8mb4_0900_ai_ci	112.0 Kib

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

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

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

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

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

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



3. Simpan kode program Langkah 2, dan jalankan perintah **php artisan migrate**. Amati apa yang terjadi pada database.
4. Buat table *database* dengan *migration* untuk table-tabel yang memiliki *foreign key*

m_barang
t_penjualan
t_stok
t_penjualan_detail

m_barang

```
Minggu_3 > Jobsheet3 > PWL_POS > database > migrations > 2025_03_04_150709_create_m_barang_table.php > class > up > Closure
1  <?php
2
3  use Illuminate\Database\Schema\Blueprint;
4  use Illuminate\Support\Facades\Schema;
5
6
7  return new class extends Migration
8  {
9
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('m_barang', function (Blueprint $table) {
16             $table->id('barang_id');
17             $table->unsignedBigInteger('kategori_id'); // Foreign Key ke m_kategori
18             $table->string('barang_kode', 10)->unique();
19             $table->string('barang_nama', 100);
20             $table->integer('harga_beli');
21             $table->integer('harga_jual');
22             $table->timestamps();
23
24             // Foreign Key
25             $table->foreign('kategori_id')->references('kategori_id')->on('m_kategori');
26         });
27
28     /**
29      * Reverse the migrations.
30      */
31     public function down(): void
32     {
33         Schema::dropIfExists('m_barang');
34     }
35 };

```

t_penjualan

```
Minggu_3 > Jobsheet3 > PWL_POS > database > migrations > 2025_03_04_150832_create_t_penjualan_table.php > class > up > Closure
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('t_penjualan', function (Blueprint $table) {
16             $table->id('penjualan_id');
17             $table->unsignedBigInteger('user_id'); // Foreign Key ke m_user
18             $table->string('pembeli', 50);
19             $table->string('penjualan_kode', 20)->unique();
20             $table->dateTime('penjualan_tanggal');
21             $table->timestamps();
22
23             // Foreign Key
24             $table->foreign('user_id')->references('user_id')->on('m_user');
25         });
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('t_penjualan');
33     }
34 };

```



t_stok

```
Minggu_3 > Jobsheet3 > PWL_POS > database > migrations > 2025_03_04_150923_create_t_stok_table.php > class > up > Closure
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('t_stok', function (Blueprint $table)
15         {
16             $table->id('stok_id');
17             $table->unsignedBigInteger('barang_id'); // Foreign Key ke m_barang
18             $table->unsignedBigInteger('user_id'); // Foreign Key ke m_user
19             $table->date('stok_tanggal');
20             $table->integer('stok_jumlah');
21             $table->timestamps();
22
23             // Foreign Key
24             $table->foreign('barang_id')->references('barang_id')->on('m_barang');
25             $table->foreign('user_id')->references('user_id')->on('m_user');
26         });
27
28         /**
29          * Reverse the migrations.
30          */
31     public function down(): void
32     {
33         Schema::dropIfExists('t_stok');
34     }
35 };

```

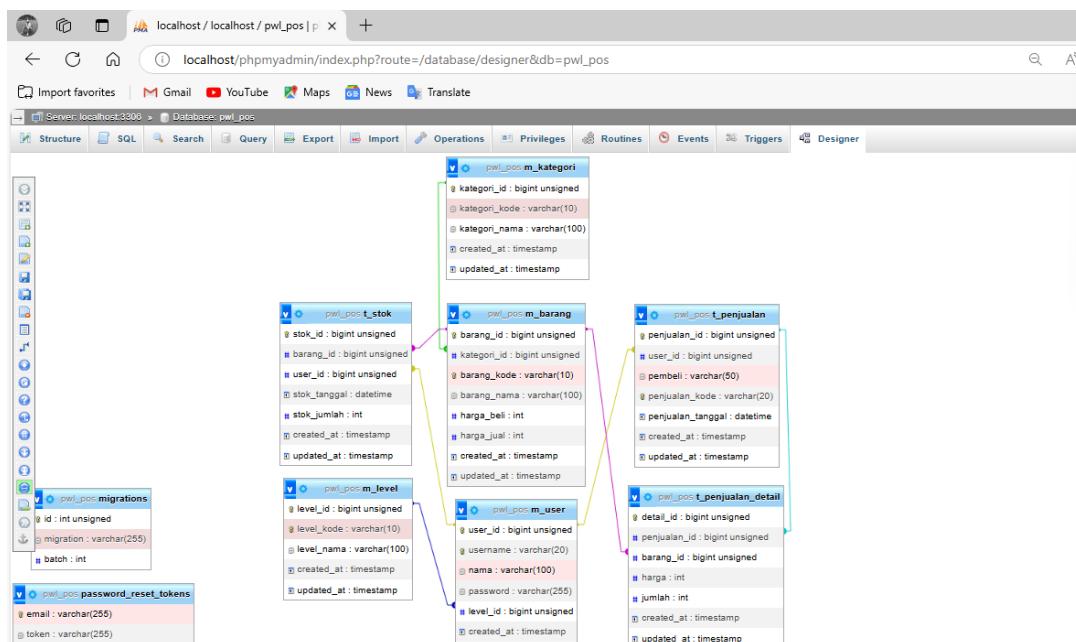
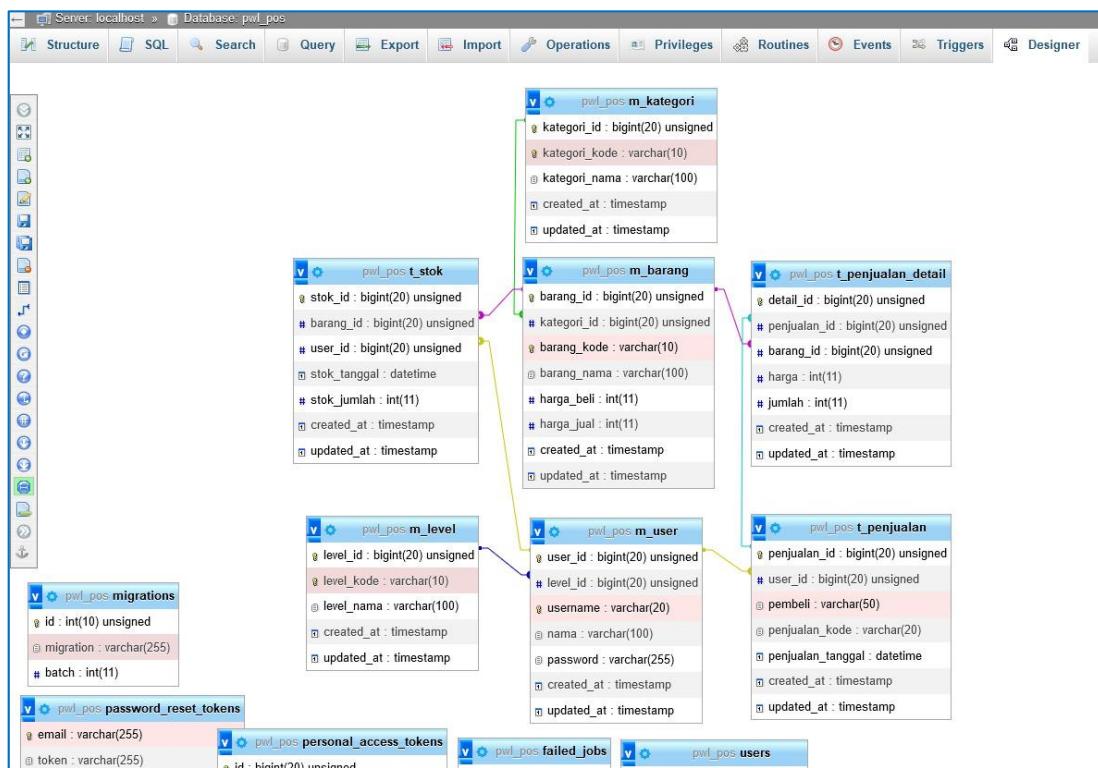
t_penjualan_detail

```
Minggu_3 > Jobsheet3 > PWL_POS > database > migrations > 2025_03_04_151009_create_t_penjualan_detail_table.php > class > up > Closure
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('t_penjualan_detail', function (Blueprint $table)
15         {
16             $table->id('detail_id');
17             $table->unsignedBigInteger('penjualan_id'); // Foreign Key ke t_penjualan
18             $table->unsignedBigInteger('barang_id'); // Foreign Key ke m_barang
19             $table->integer('harga');
20             $table->integer('jumlah');
21             $table->timestamps();
22
23             // Foreign Key
24             $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan');
25             $table->foreign('barang_id')->references('barang_id')->on('m_barang');
26         });
27
28         /**
29          * Reverse the migrations.
30          */
31     public function down(): void
32     {
33         Schema::dropIfExists('t_penjualan_detail');
34     }
35 };

```

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/seeders**

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

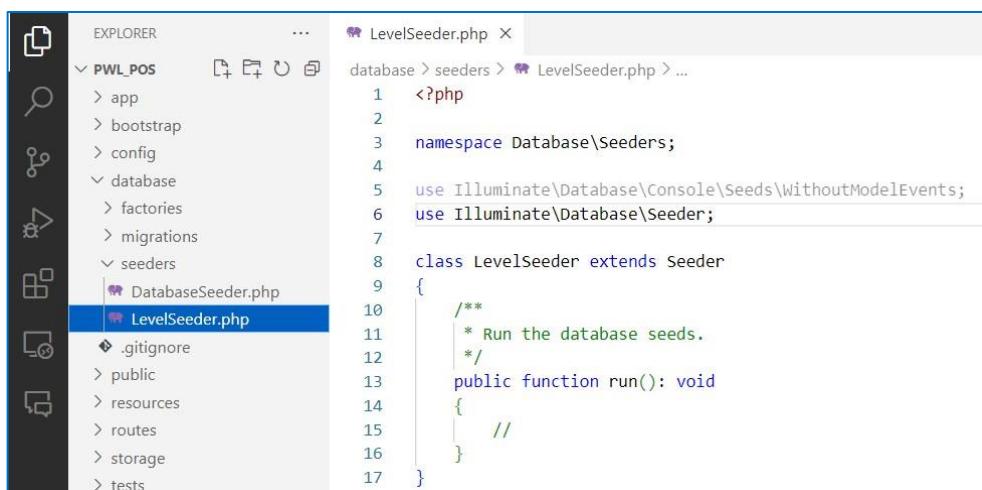
```
php artisan db:seed --class=<nama-class-seeder>
```

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
```



```
EXPLORER      ...      LevelSeeder.php ×  
  PWL_POS      D+ D- ⌂ ⌂  
    > app  
    > bootstrap  
    > config  
    > database  
    > factories  
    > migrations  
    > seeders  
      DatabaseSeeder.php  
      LevelSeeder.php  
    .gitignore  
    > public  
    > resources  
    > routes  
    > storage  
    > tests  
  
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   * 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);
    }
}
```

```
<?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

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

```
PS D:\_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
<input type="checkbox"/>		Edit		Copy		Delete	2 MNG
<input type="checkbox"/>		Edit		Copy		Delete	3 STF

Check all With selected: Edit Copy Delete Export

performance_schema

pwl_pos	New	failed_jobs	migrations	m_barang	m_kategori	m_level	m_user	nasabah	reset_tokens
---------	-----	-------------	------------	----------	------------	----------------	--------	---------	--------------

Extra options

	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	NULL	NULL
<input type="checkbox"/>		Edit		Copy		Delete	5 CUS	Pelanggan	2025-03-05 02:38:33	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 }
```

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Seeder;
6 use Illuminate\Support\Facades\DB;
7 use Illuminate\Support\Facades\Hash;
8
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.LySwH.Y.4oAKU7FzwS6fW...
<input type="checkbox"/>		2	2	manager	Manager	\$2y\$12\$Ajfn20/FdPTeUgghz31muEhIxFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/>		3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMy/BHnbJ9W...

		user_id	username	nama	password	level_id	created_at	updated_at
<input type="checkbox"/>		1	admin	Administrator	\$2y\$12\$GfVlhOuWaTiduoLMAfomWOU0Uj2hBufnzDbTRfWix...	1	NULL	NULL
<input type="checkbox"/>		2	manager	Manager	\$2y\$12\$DS23lOTzLnIMKRm0f050e8yRCYuvrHuZN1bqx4J8Z...	2	NULL	NULL
<input type="checkbox"/>		3	staff	Staff/Kasir	\$2y\$12\$eB/Wmtc7J0syRkp2vuS42DG9oddv0pjBEQoYAK...	3	NULL	NULL
<input type="checkbox"/>		18	customer-1	Pelanggan Pertama	\$2y\$12\$TsByA3ZYxDR2WuEjhHONIBcB.206mLuEZgv5gSH...	5	NULL	2025-03-05 02:42:14

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

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

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

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

m_kategori								
		kategori_id	kategori_kode	kategori_nama	created_at	updated_at	Extra options	
<input type="checkbox"/>		1	KTN001	Alat Tulis	NULL	NULL		
<input type="checkbox"/>		2	KTN002	Pakaian	NULL	NULL		
<input type="checkbox"/>		3	KTN003	Peralatan Rumah	NULL	NULL		
<input type="checkbox"/>		4	KTN004	Makanan	NULL	NULL		
<input type="checkbox"/>		5	KTN005	Minuman	NULL	NULL		

m_barang								
		barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at
<input type="checkbox"/>		1	1	BRG001	Keripik Kentang	4500	7000	NULL
<input type="checkbox"/>		2	1	BRG002	Donat Coklat	6000	8500	NULL
<input type="checkbox"/>		3	2	BRG003	Susu Kotak	3500	5000	NULL
<input type="checkbox"/>		4	2	BRG004	Mie Instan	4000	6500	NULL
<input type="checkbox"/>		5	3	BRG005	Headset Bluetooth	120000	180000	NULL
<input type="checkbox"/>		6	3	BRG006	Monitor LED	300000	450000	NULL
<input type="checkbox"/>		7	4	BRG007	Keset Karet	18000	25000	NULL
<input type="checkbox"/>		8	4	BRG008	Ember Plastik	25000	35000	NULL
<input type="checkbox"/>		9	5	BRG009	Celana Jeans	60000	75000	NULL
<input type="checkbox"/>		10	5	BRG010	Sweater Rajut	120000	180000	NULL



t_stok

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	1	1	3	2025-03-05 01:23:22	50	NULL	NULL
<input type="checkbox"/>	2	2	3	2025-03-05 01:23:22	40	NULL	NULL
<input type="checkbox"/>	3	3	3	2025-03-05 01:23:22	60	NULL	NULL
<input type="checkbox"/>	4	4	3	2025-03-05 01:23:22	30	NULL	NULL
<input type="checkbox"/>	5	5	3	2025-03-05 01:23:22	70	NULL	NULL
<input type="checkbox"/>	6	6	3	2025-03-05 01:23:22	45	NULL	NULL
<input type="checkbox"/>	7	7	3	2025-03-05 01:23:22	35	NULL	NULL
<input type="checkbox"/>	8	8	3	2025-03-05 01:23:22	55	NULL	NULL
<input type="checkbox"/>	9	9	3	2025-03-05 01:23:22	25	NULL	NULL
<input type="checkbox"/>	10	10	3	2025-03-05 01:23:22	65	NULL	NULL

t_penjualan

<input type="checkbox"/>	1	3	Pelanggan 1	PNJ0001	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	2	3	Pelanggan 2	PNJ0002	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	3	3	Pelanggan 3	PNJ0003	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	4	3	Pelanggan 4	PNJ0004	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	5	3	Pelanggan 5	PNJ0005	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	6	3	Pelanggan 6	PNJ0006	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	7	3	Pelanggan 7	PNJ0007	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	8	3	Pelanggan 8	PNJ0008	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	9	3	Pelanggan 9	PNJ0009	2025-03-05 01:20:37	NULL	NULL
<input type="checkbox"/>	10	3	Pelanggan 10	PNJ0010	2025-03-05 01:20:37	NULL	NULL

t_penjualan_detail

<input type="checkbox"/>	1	1	1	5000	2	NULL	NULL
<input type="checkbox"/>	2	1	2	7000	1	NULL	NULL
<input type="checkbox"/>	3	1	3	3000	3	NULL	NULL
<input type="checkbox"/>	4	2	4	4000	2	NULL	NULL
<input type="checkbox"/>	5	2	5	150000	1	NULL	NULL
<input type="checkbox"/>	6	2	6	350000	1	NULL	NULL
<input type="checkbox"/>	7	3	7	20000	1	NULL	NULL
<input type="checkbox"/>	8	3	8	25000	2	NULL	NULL
<input type="checkbox"/>	9	3	9	50000	1	NULL	NULL
<input type="checkbox"/>	10	4	10	120000	1	NULL	NULL
<input type="checkbox"/>	11	4	1	5000	3	NULL	NULL
<input type="checkbox"/>	12	4	2	7000	1	NULL	NULL
<input type="checkbox"/>	13	5	3	3000	2	NULL	NULL
<input type="checkbox"/>	14	5	4	4000	1	NULL	NULL
<input type="checkbox"/>	15	5	5	150000	1	NULL	NULL
<input type="checkbox"/>	16	6	6	350000	1	NULL	NULL
<input type="checkbox"/>	17	6	7	20000	2	NULL	NULL
<input type="checkbox"/>	18	6	8	25000	1	NULL	NULL
<input type="checkbox"/>	19	7	9	50000	3	NULL	NULL
<input type="checkbox"/>	20	7	10	120000	1	NULL	NULL

<input type="checkbox"/>	21	7	1	5000	2	NULL	NULL
<input type="checkbox"/>	22	8	2	7000	2	NULL	NULL
<input type="checkbox"/>	23	8	3	3000	1	NULL	NULL
<input type="checkbox"/>	24	8	4	4000	3	NULL	NULL
<input type="checkbox"/>	25	9	5	150000	1	NULL	NULL

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>	26	9	6	350000	1	NULL	NULL
<input type="checkbox"/>	27	9	7	20000	2	NULL	NULL
<input type="checkbox"/>	28	10	8	25000	1	NULL	NULL
<input type="checkbox"/>	29	10	9	50000	3	NULL	NULL
<input type="checkbox"/>	30	10	10	120000	1	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
```

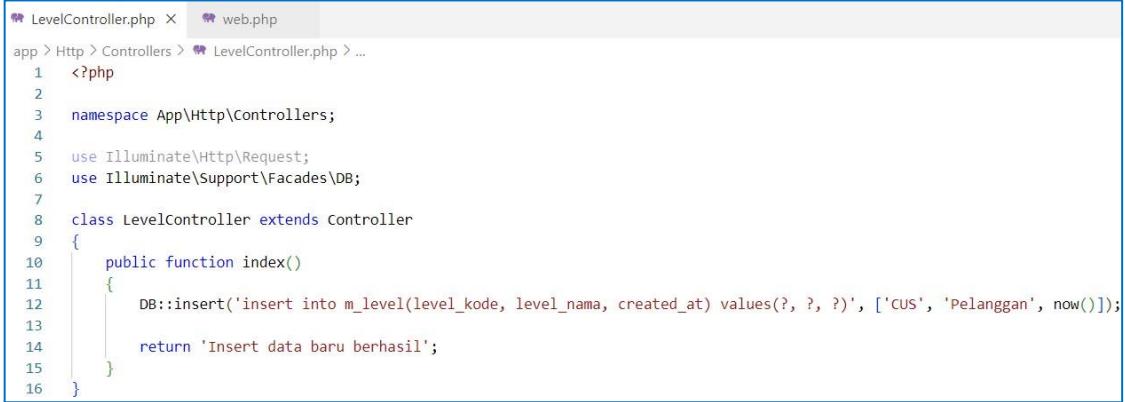
2. Kita modifikasi dulu untuk *routing*-nya, ada di [PWL_POS/routes/web.php](#)

```
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']);
```

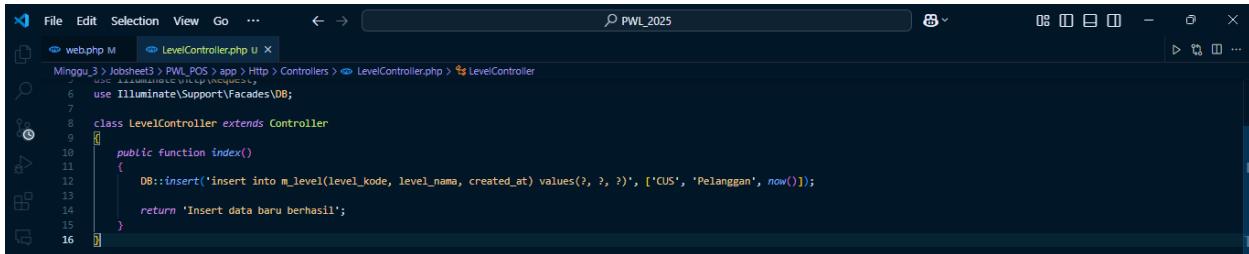
```
Minggu_3 > Jobsheet3 > PWL_POS > routes > web.php > ...
12 /**
13  * Here is where you can register web routes for your application. These
14  * routes are loaded by the RouteServiceProvider and all of them will
15  * be assigned to the "web" middleware group. Make something great!
16  */
17
18 //Route
19 Route::get('/welcome', function () {
20     |     return view('welcome');
21 });
22
23 Route::get('/level', [LevelController::class, 'index']);
24 Route::get('/kategorien', [KategorieController::class, 'index']);
```



3. Selanjutnya, kita modifikasi file [LevelController](#) untuk menambahkan 1 data ke table [m_level](#)

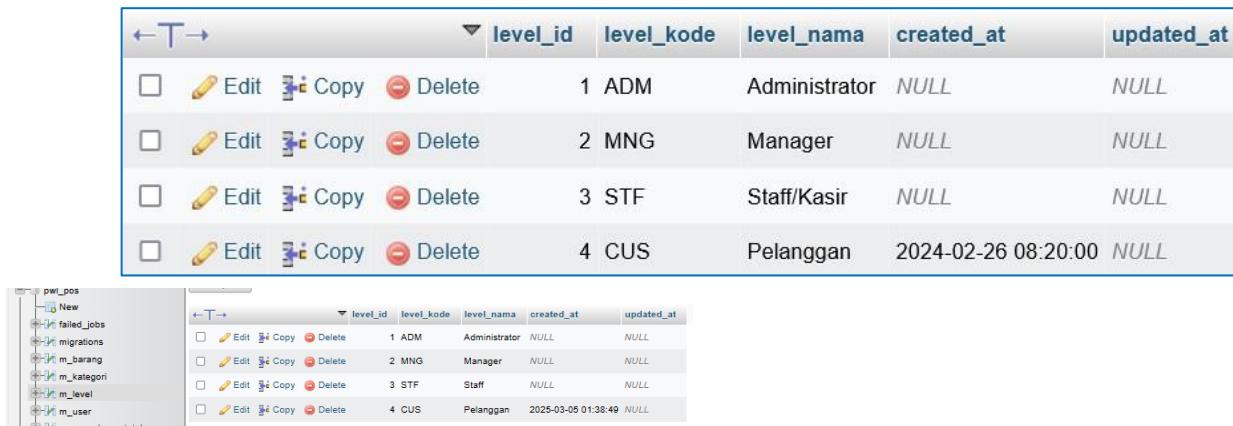


```
LevelController.php × web.php
app > Http > Controllers > LevelController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12        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



```
File Edit Selection View Go ... ← → ⌂ PWL_2025
web.php M LeveController.php u X
Minggu_3 > Jobsheet3 > PWL_POS > app > Http > Controllers > LeveController.php > LeveController
use Illuminate\Support\Facades\DB;
class LeveController extends Controller
{
    public function index()
    {
        DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
        return 'Insert data baru berhasil';
    }
}
```

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`

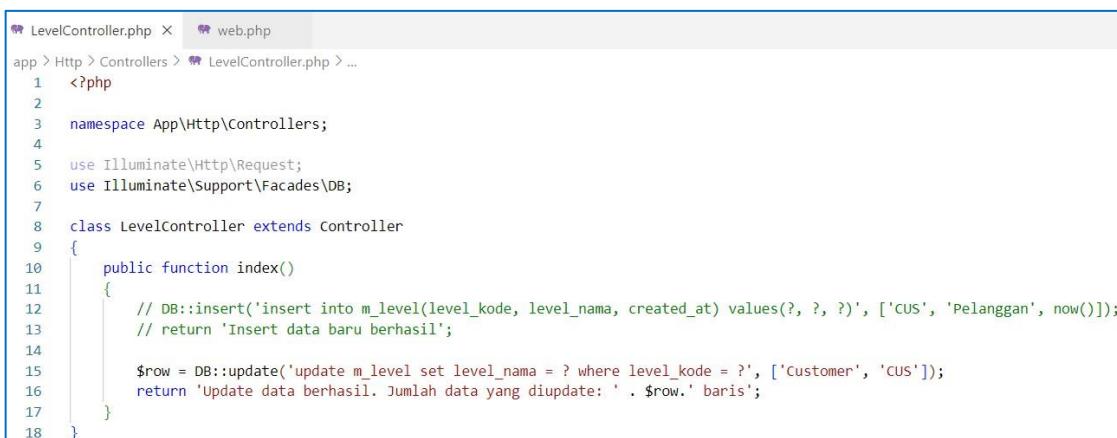


	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit Copy Delete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

pwl_pos

- New
- failed_jobs
- migrations
- m_barang
- m_kategori
- m_level**
- m_user
- password_reset_tokens

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



```
LeveController.php x web.php
app > Http > Controllers > LeveController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LeveController 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`



		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	NULL	NULL
<input type="checkbox"/>		4	CUS	Customer	2025-03-05 01:38:49	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 }
```



The screenshot shows the MySQL Workbench interface with the 'm_level' table selected. The table has columns: level_id, level_kode, level_nama, created_at, and updated_at. The data shows three rows: ADM (Administrator), MNG (Manager), and STF (Staff).

	level_id	level_kode	level_nama	created_at	updated_at
1	ADM	Administrator	NULL	NULL	
2	MNG	Manager	NULL	NULL	
3	STF	Staff	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/views/level.blade.php**

The screenshot shows the code for the **level.blade.php** file. It contains an HTML structure with a title and a table to display data from the **m_level** table.

```
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

Akan muncul tabel dari level pengguna

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

Praktikum 5 – Implementasi *Query Builder*

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

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

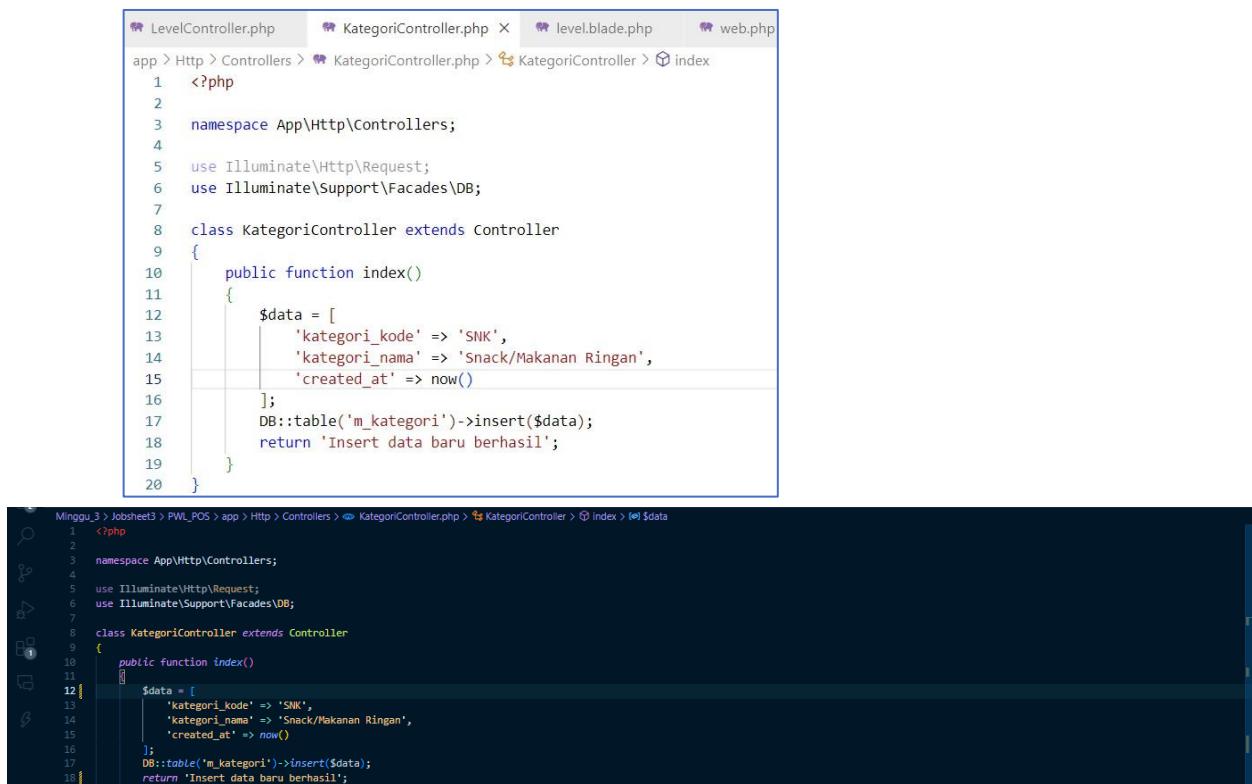
2. Kita modifikasi dulu untuk routing-nya, ada di `PWL_POS/routes/web.php`

```
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']);
```

```
Minggu_3 > Jobsheet3 > PWL_POS > routes > web.php > ...
11
12  / Here is where you can register web routes for your application. These
13  / routes are loaded by the RouteServiceProvider and all of them will
14  / be assigned to the "web" middleware group. Make something great!
15  /
16  /**
17   * @Route
18   */
19  Route::get('/welcome', function () {
20      return view('welcome');
21  });
22
23  Route::get('/level', [LevelController::class, 'index']);
24  Route::get('/kategori', [KategoriController::class, 'index']);
25  Route::get('/user', [UserController::class, 'index']);
```



3. Selanjutnya, kita modifikasi file [KategoriController](#) untuk menambahkan 1 data ke table [m_kategori](#)



```
LevelController.php  KategoriController.php  level.blade.php  web.php
app > Http > Controllers > KategoriController.php > 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 }
```

```
Minggu_3 > Jobsheet3 > PWL_POS > app > Http > Controllers > KategoriController.php > index > $data
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class KategoriController extends Controller
9 {
10     public function index()
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil';
19     }
20 }
```

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



	kategori_id	kategori.kode	kategori.nama	created_at	updated_at
	1	KTN001	Alat Tulis	NULL	NULL
	2	KTN002	Pakaian	NULL	NULL
	3	KTN003	Peralatan Rumah	NULL	NULL
	4	KTN004	Makanan	NULL	NULL
	5	KTN005	Minuman	NULL	NULL
	6	SNK	Snack/Makanan Ringan	2025-03-05 01:52:55	NULL

Terdapat value baru yaitu snack/makanan ringan yang ditambahkan melalui route.



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 }
```

19 \$row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
20 return 'Update data berhasil. Jumlah data yang diupdate: ' . \$row . ' baris';
21
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**

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
	1	KTN001	Alat Tulis	NULL	NULL
	2	KTN002	Pakaian	NULL	NULL
	3	KTN003	Peralatan Rumah	NULL	NULL
	4	KTN004	Makanan	NULL	NULL
	5	KTN005	Minuman	NULL	NULL
	6	SNK	Camilan	2025-03-05 01:52:55	NULL

Melakukan perubahan di **kategori_nama** menjadi **camilan**.

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
21
22
23
24
25     */
}
```

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
	1	KTN001	Alat Tulis	NULL	NULL
	2	KTN002	Pakaian	NULL	NULL
	3	KTN003	Peralatan Rumah	NULL	NULL
	4	KTN004	Makanan	NULL	NULL
	5	KTN005	Minuman	NULL	NULL

Value sebelumnya dihapus dari database.

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
21
22
23
24
25
26
27
28     */
}
26
27
28     $data = DB::table('m_kategori')->get();
return view('kategori', ['data' => $data]);
```

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	KTN001	Alat Tulis
2	KTN002	Pakaian
3	KTN003	Peralatan Rumah
4	KTN004	Makanan
5	KTN005	Minuman

Terdapat tabel yang menampilkan kode barang dan nama kategori yang diambil dari database.

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

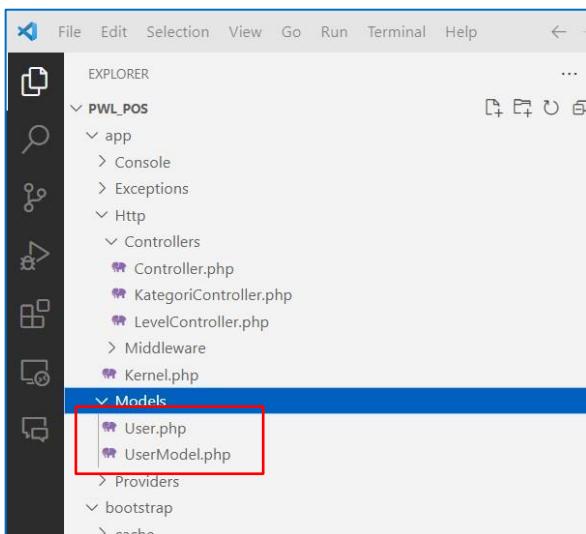
```
php artisan make:model <nama-model-CamelCase>
```

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
```





```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class UserModel extends Model
{
    use HasFactory;
}
protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
```

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

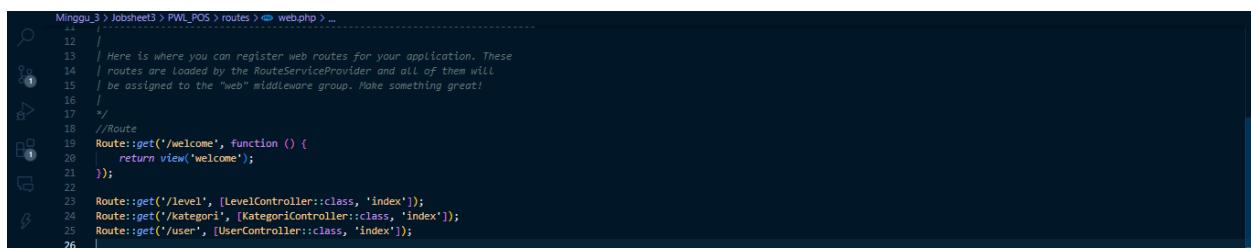
```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class UserModel extends Model
{
    use HasFactory;
}
protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
```

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class UserModel extends Model
{
    use HasFactory;
}
protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
```



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

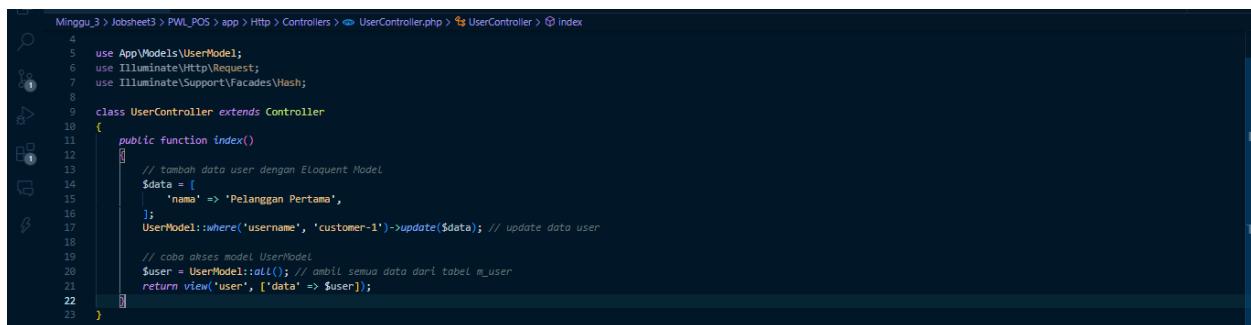
```
routes > 📄 web.php > ...
1   <?php
2
3   use App\Http\Controllers\KategoriController;
4   use App\Http\Controllers\LevelController;
5   use App\Http\Controllers\UserController;
6   use Illuminate\Support\Facades\Route;
7
8
9   Route::get('/', function () {
10     return view('welcome');
11   });
12
13   Route::get('/level', [LevelController::class, 'index']);
14   Route::get('/kategori', [KategoriController::class, 'index']);
15   Route::get('/user', [UserController::class, 'index']);
16
```



```
Minggu_3 > Jobsheet3 > PWL_POS > routes > 📄 web.php > ...
12  /
13  / Here is where you can register web routes for your application. These
14  / routes are loaded by the RouteServiceProvider and all of them will
15  / be assigned to the "web" middleware group. Make something great!
16  /
17  */
18
19 //Route
20 Route::get('/welcome', function () {
21   return view('welcome');
22 });
23
24 Route::get('/level', [LevelController::class, 'index']);
25 Route::get('/kategori', [KategoriController::class, 'index']);
26 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 }
```



```
Minggu_3 > Jobsheet3 > PWL_POS > app > Http > Controllers > 📄 UserController.php > UserController > ⏺ index
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      $data = [
15        'nama' => 'Pelanggan Pertama',
16      ];
17      UserModel::where('username', 'customer-1')->update($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 }
```



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>

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

Menampilkan data user dengan informasi username, nama, dan id level pengguna.

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

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
16	customer-1	Pelanggan	5

Menampilkan data user dengan user baru yaitu Pelanggan.

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
16	customer-1	Pelanggan Pertama	5

Menampilkan semua user dengan user baru yaitu customer dengan nama pelanggan pertama.

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?

`APP_KEY` digunakan untuk kunci enkripsi di laravel. Fungsinya adalah untuk mengenkripsi data sensitif, seperti session pengguna dan token API, serta memastikan keamanan aplikasi.

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

Menggunakan perintah 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?

Saat membuat proyek Laravel baru, Laravel secara default memiliki 3 file migrasi, yaitu :

- `create_users_table.php` yang berfungsi untuk membuat tabel users untuk menyimpan data pengguna.
- `create_password_resets_table.php` yang berfungsi untuk menyimpan token reset password.
- `create_failed_jobs_table.php` yang berfungsi untuk menyimpan informasi tentang job yang gagal dijalankan di queue system.

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

`$table->timestamps();` digunakan untuk otomatis membuat 2 kolom di tabel database :

- `created_at` yang digunakan untuk menyimpan waktu saat data dibuat.
- `updated_at` yang digunakan untuk menyimpan waktu saat data diperbarui.



5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
`$table->id();` digunakan untuk membuat kolom primary key dengan tipe data BIGINT UNSIGNED
6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?
`$table->id();` secara default akan membuat kolom id sebagai primary key.
`$table->id('level_id');` akan membuat kolom level_id sebagai primary key, bukan id.
7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
`->unique()` digunakan untuk memastikan nilai dalam kolom tersebut tidak 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')` ?
Di `m_level`, `level_id` dibuat menggunakan `$table->id('level_id')`, yang otomatis menjadi primary key dengan tipe BIGINT UNSIGNED AUTO_INCREMENT.
Di `m_user`, `level_id` harus sama tipe datanya dengan `level_id` di `m_level`, yaitu BIGINT UNSIGNED, jadi digunakan `$table->unsignedBigInteger('level_id')`.
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?
Class Hash digunakan untuk mengenkripsi password agar tidak tersimpan dalam bentuk teks biasa di database. `Hash::make('1234');` akan mengubah '1234' menjadi format hash
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?



Tanda tanya ? digunakan sebagai pengganti dalam query, yang mencegah 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'; memberitahu laravel bahwa model ini terkait dengan tabel m_user, bukan users (default laravel).
protected \$primaryKey = 'user_id'; menentukan bahwa primary key tabel ini adalah user_id, bukan id (default laravel).
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan
Menurut saya lebih mudah menggunakan Query Builder karena kita tidak perlu menuliskan sintaks sql nya secara langsung. Hal ini memudahkan saya dalam hal efisiensi waktu pengerjaan

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