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

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.**

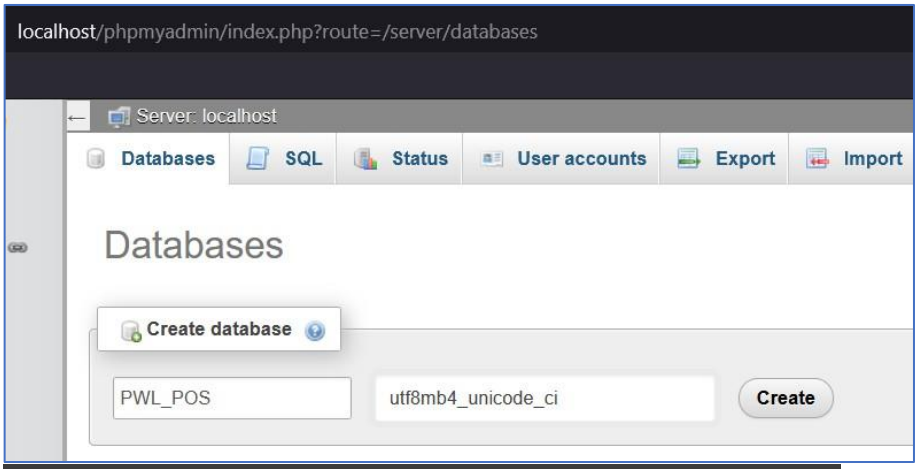
Project **PWL_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajarai

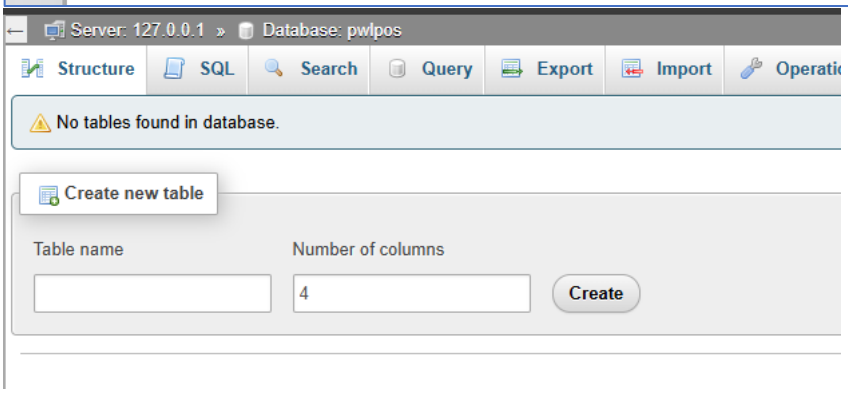
A. PENGATURAN DATABASE

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

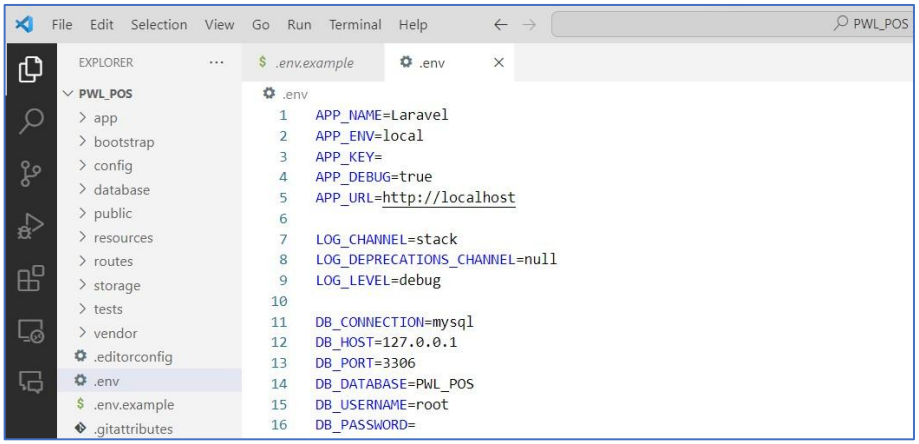
Praktikum 1 - pengaturan database:

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

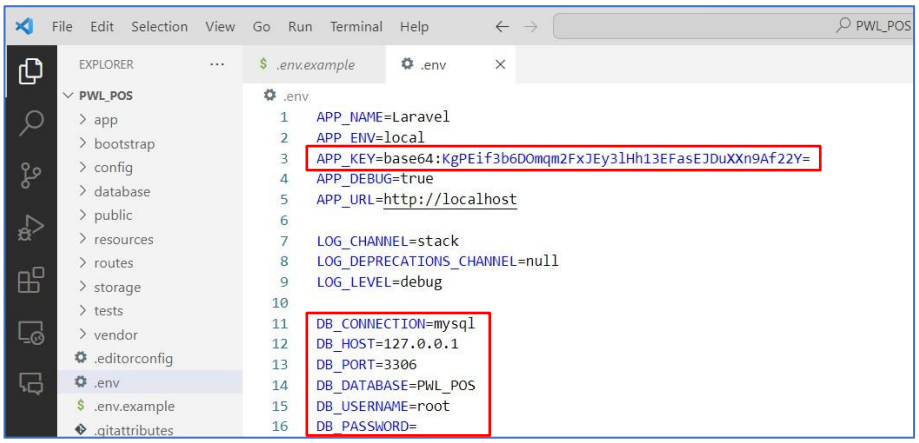


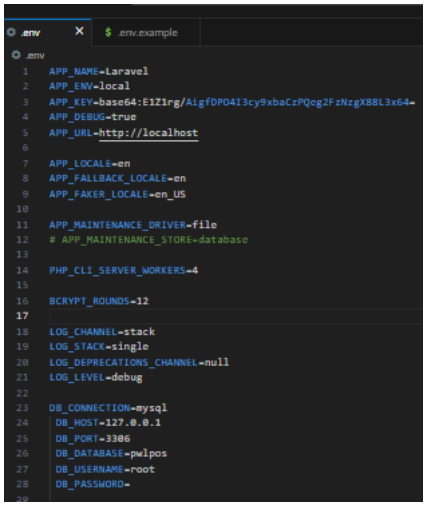


2. Buka aplikasi VSCode dan buka folder project `PWL_POS` yang sudah kita buat
3. Copy file `.env.example` menjadi `.env`
4. Buka file `.env`, dan pastikan konfigurasi **`APP_KEY`** bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan `php artisan`.



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





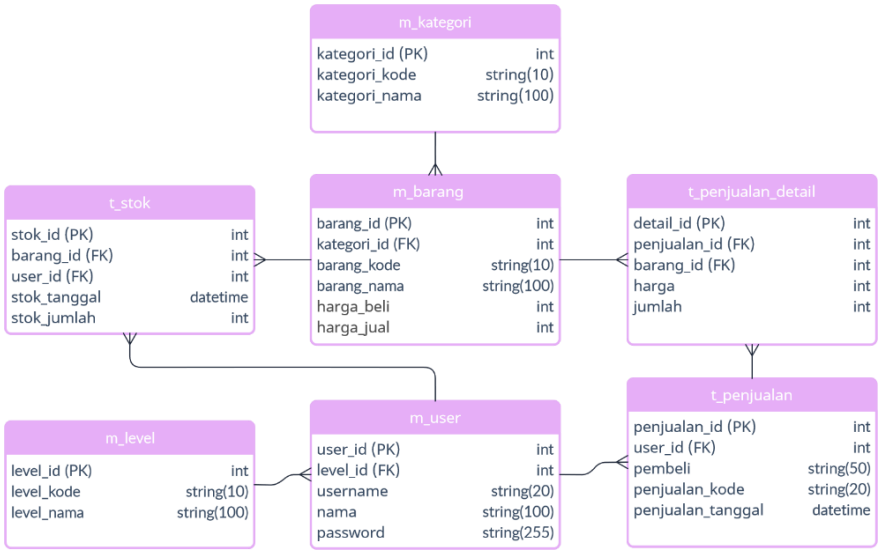
6. Laporkan hasil Praktikum-1 ini dan *commit* perubahan pada *git*.

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

INFO

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

Pembuatan file migrasi bisa menggunakan 2 cara, yaitu

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

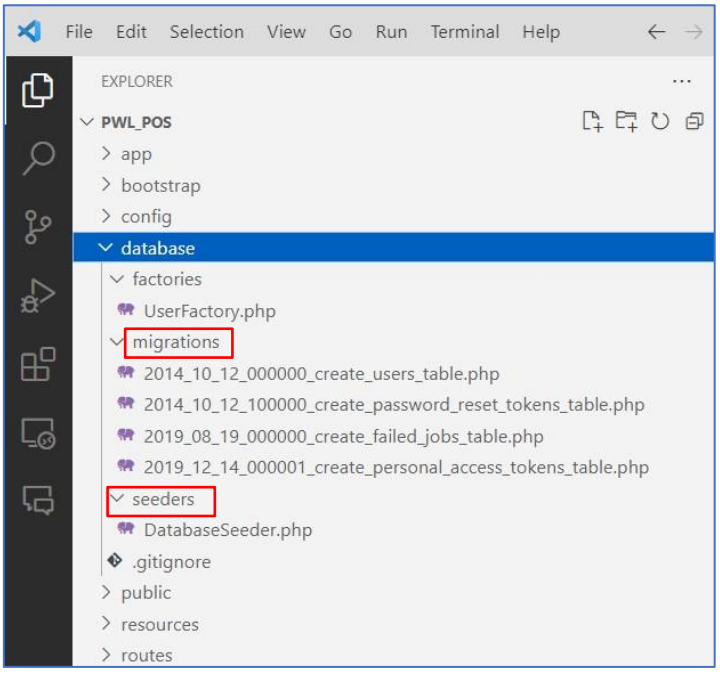
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

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

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

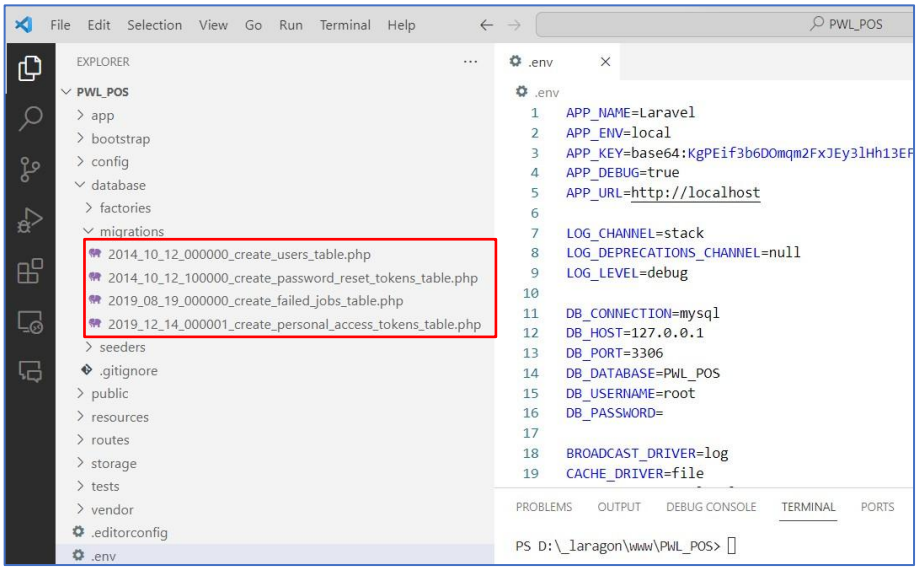
Perintah **`-m`** di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

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



Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



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

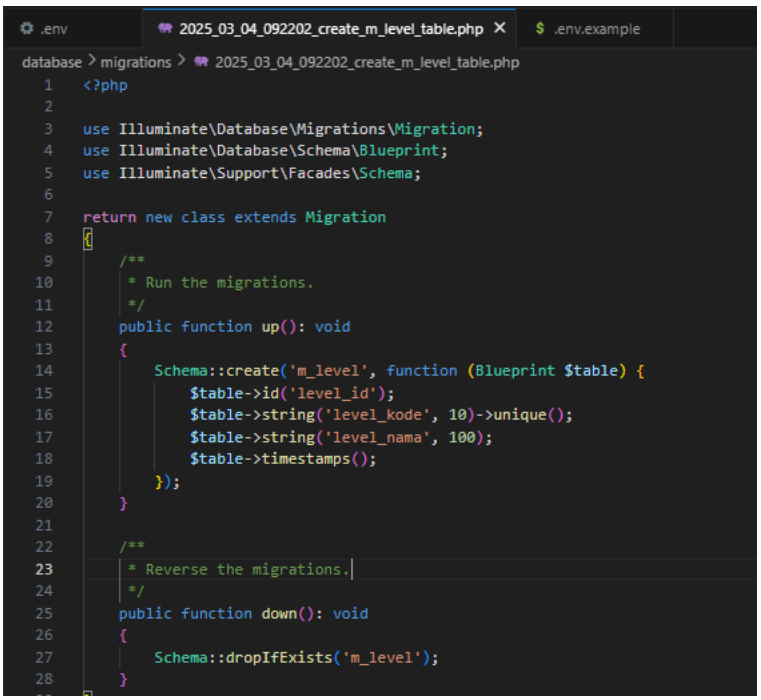
```
php artisan make:migration create_m_level_table --create=m_level
```


```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:migration create_m_level_table
--create=m_level
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_level', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_level');
    }
};
```

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

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    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();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_level');
    }
};
```



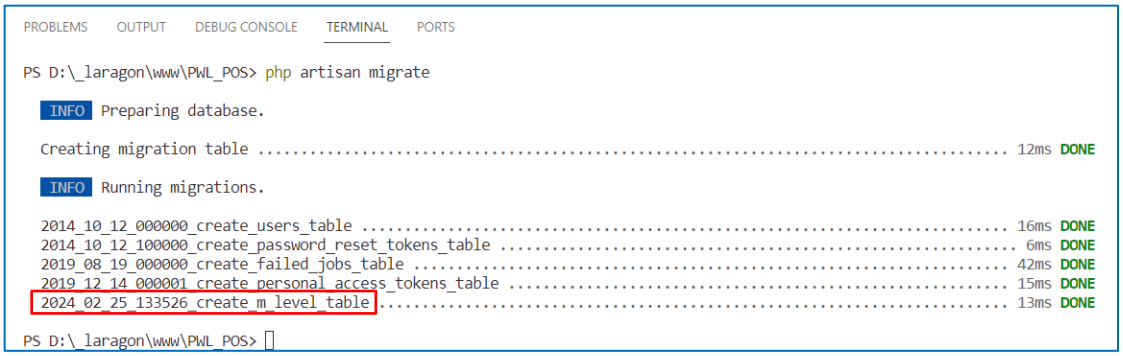
INFO

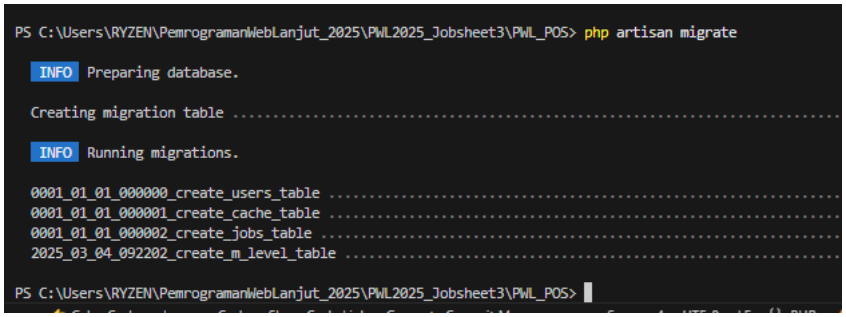
Dalam fitur migration Laravel, terdapat berbagai macam function untuk membuat kolom di table database. Silahkan cek disini

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

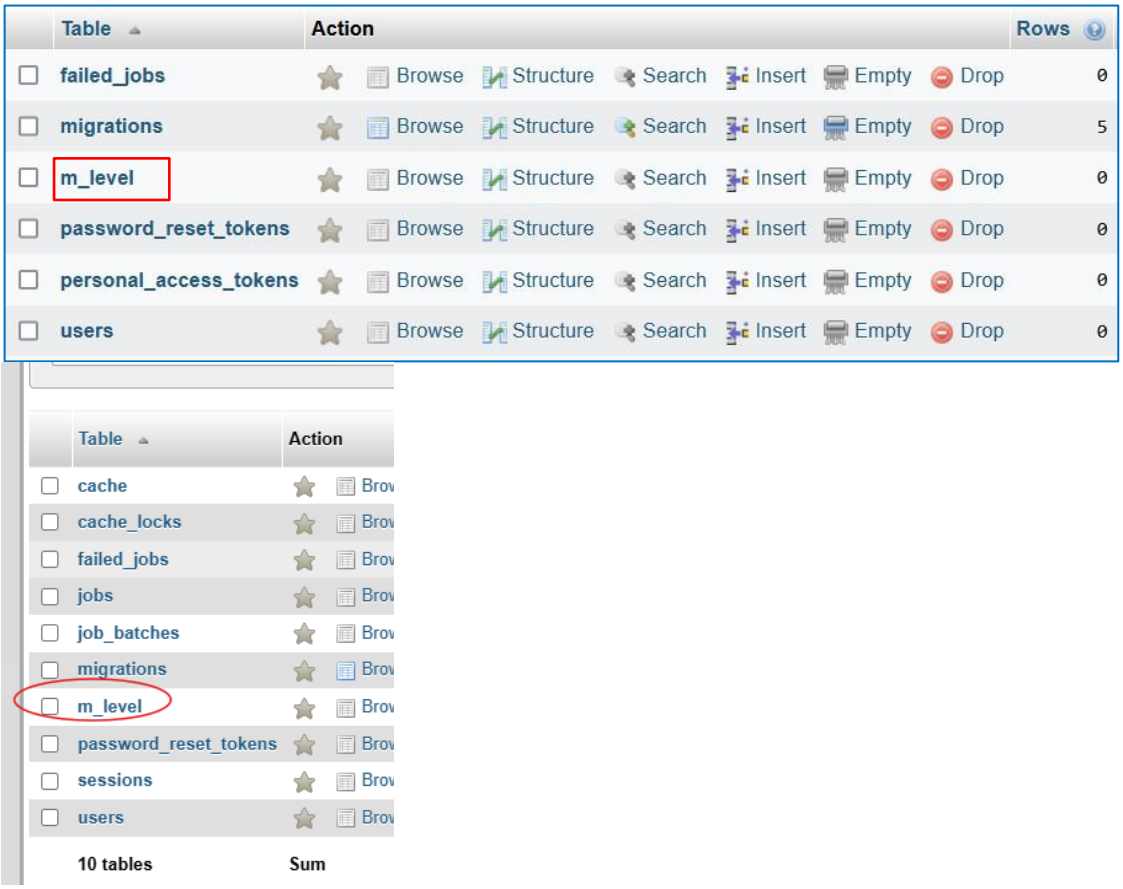
5. Simpan kode pada tahapan 4 tersebut, kemudian jalankan perintah ini pada terminal VSCode untuk melakukan migrasi

```
php artisan migrate
```

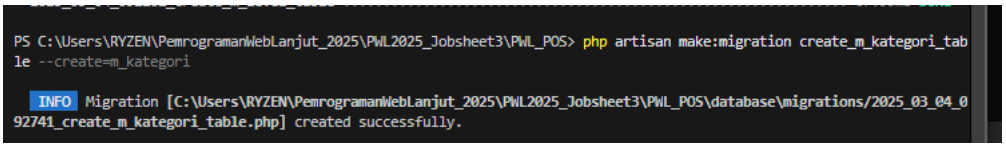




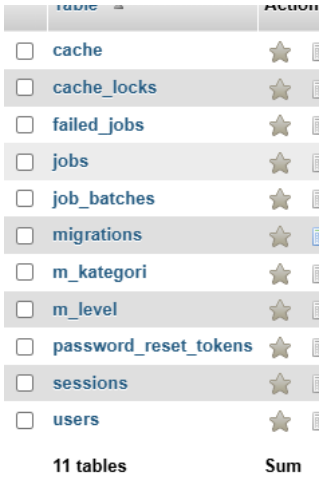
6. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum



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



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_kategori', function (Blueprint $table) {
            $table->id('kategori_id');
            $table->string('kategori_kode', 10)->unique();
            $table->string('kategori_nama', 100);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_kategori');
    }
};
```



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
```

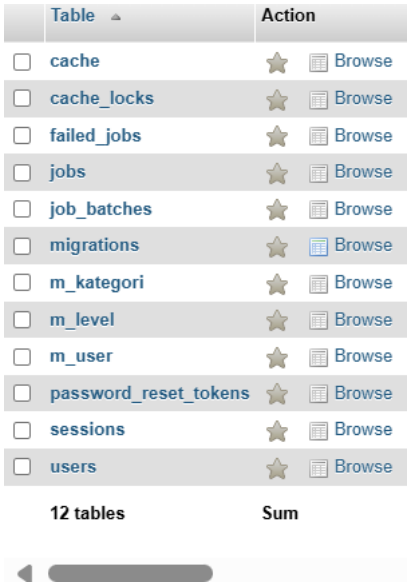
```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:migration create_m_user_table --create=m_user

  INFO  Migration [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\migrations/2025_03_04_093051_create_m_user_table.php] created successfully.

PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS>
```

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

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_user', function (Blueprint $table) {
            $table->id('user_id');
            $table->unsignedBigInteger('level_id')->index(); // indexing untuk ForeignKey
            $table->string('username', 20)->unique(); // unique untuk memastikan tidak ada username yang sama
            $table->string('nama', 100);
            $table->string('password');
            $table->timestamps();

            // Mendefinisikan Foreign Key pada kolom level_id mengacu pada kolom level_id di tabel m_level
            $table->foreign('level_id')->references('level_id')->on('m_level');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_user');
    }
};
```

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_user', function (Blueprint $table) {
            $table->id('user_id');
            $table->unsignedBigInteger('level_id')->index();
            $table->string('username', 20)->unique();
            $table->string('nama', 100);
            $table->string('password');
            $table->timestamps();


            $table->foreign('level_id')->references('level_id')->on('m_level');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_user');
    }
};
```

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 :

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:migration create_m_barang_table --create=m_barang

INFO  Migration [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\migrations/2025_03_04_093652_create_m_barang_table.php] created successfully.
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('m_barang', function (Blueprint $table) {
            $table->id('barang_id');
            $table->string('nama_barang', 100);
            $table->integer('harga');
            $table->integer('stok');
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('m_barang');
    }
};
```

T_penjualan :

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:migration create_t_penjualan_table --create=t_penjualan

  INFO  Migration [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\migrations/2025_03_04_093902_create_t_penjualan_table.php] created successfully.
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('t_penjualan', function (Blueprint $table) {
            $table->id('penjualan_id');
            $table->date('tanggal');
            $table->string('nama_pelanggan', 100);
            $table->timestamps();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('t_penjualan');
    }
};
```

T_stok :

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:migration create_t_stok_table --create=t_stok

  INFO  Migration [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\migrations/2025_03_04_094050_create_t_stok_table.php] created successfully.
```

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('t_stok', function (Blueprint $table) {
            $table->id('stok_id');
            $table->unsignedBigInteger('barang_id');
            $table->integer('jumlah');
            $table->string('keterangan', 100);
            $table->timestamps();

            // Foreign Key
            $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('t_stok');
    }
};

```

T_penjualan_detail:

```php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('t_stok', function (Blueprint $table) {
            $table->id('stok_id');
            $table->unsignedBigInteger('barang_id');
            $table->integer('jumlah');
            $table->string('keterangan', 100);
            $table->timestamps();

            // Foreign Key
            $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('t_stok');
    }
};

```

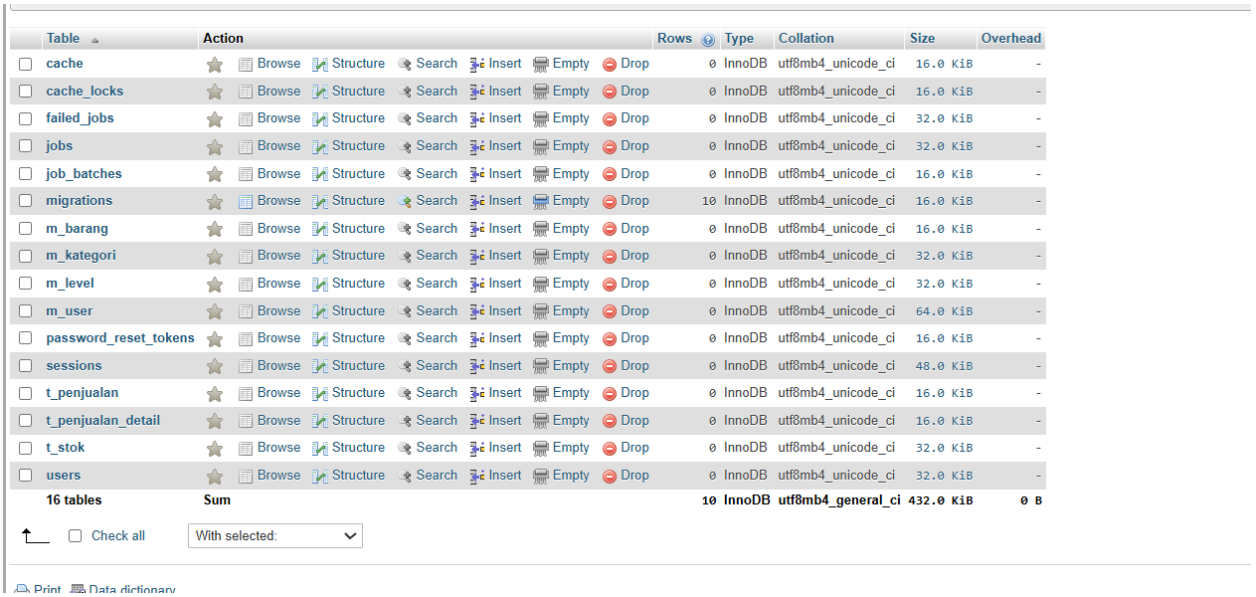
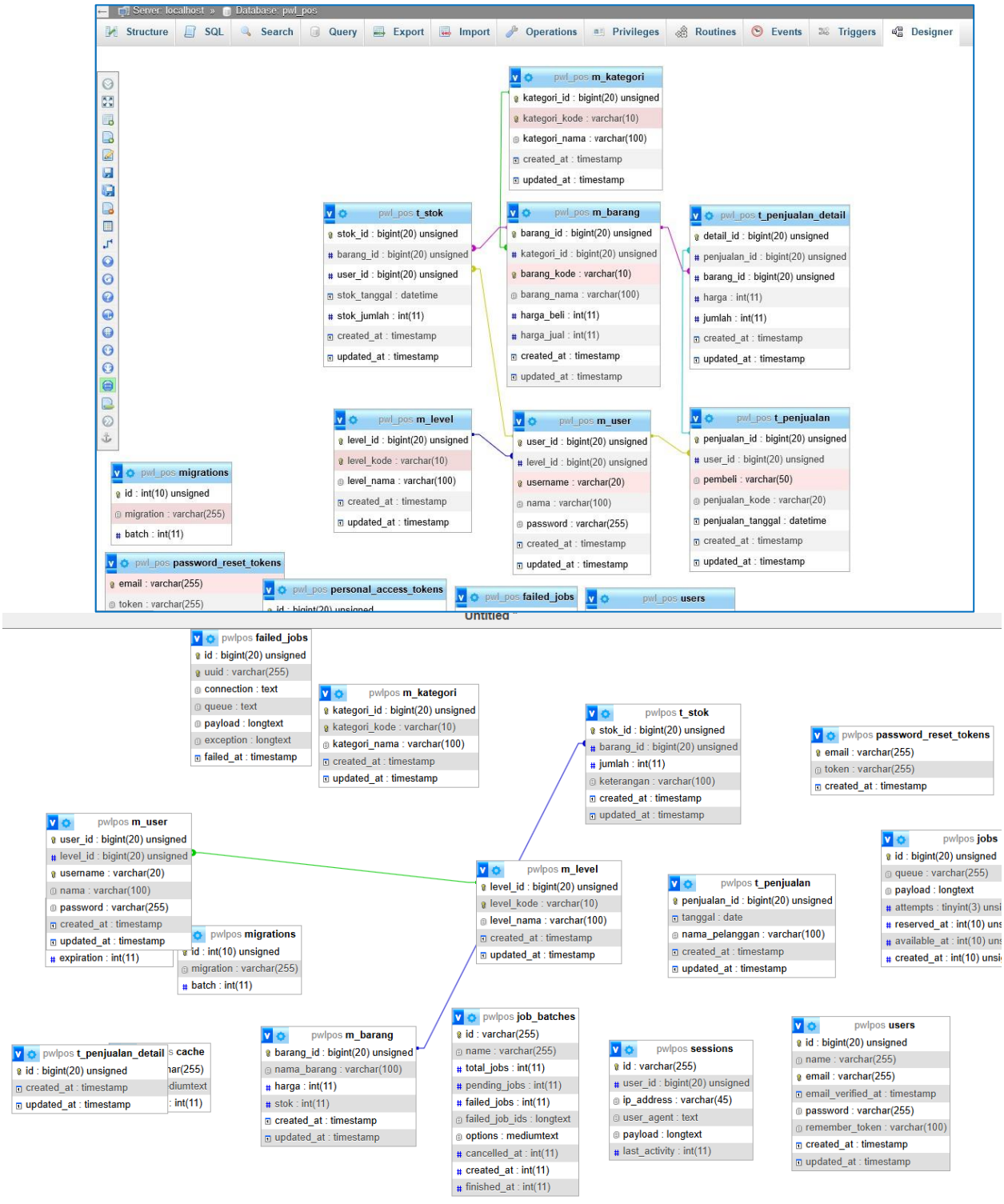


Table	Action	Rows	Type	Collation	Size	Overhead
cache	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
cache_locks	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
job_batches	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
migrations	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_barang	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_kategori	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
m_level	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
m_user	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	64.0 KiB	-
password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
sessions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
t_penjualan	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
t_penjualan_detail	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
t_stok	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
16 tables	Sum	10	InnoDB	utf8mb4_general_ci	432.0 KiB	0 B

Check all With selected:

Print Data dictionary

5. ika 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 penggunna 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

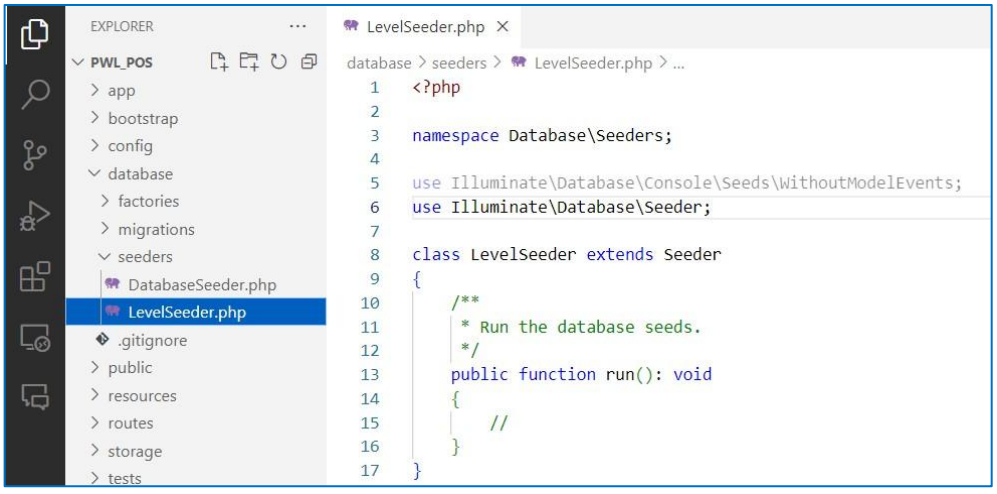
```
php artisan db:seed --class=<nama-class-seeder>
```

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
```



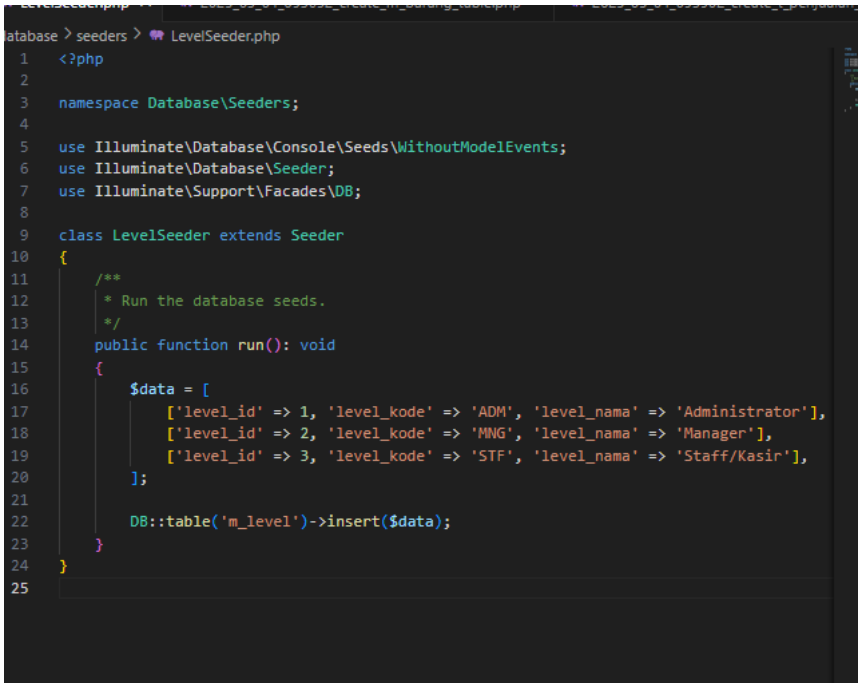
```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\P
WL_POS> php artisan make:seeder LevelSeeder
```

LevelSeeder.php | 2025_03_04_093652_create_m_barang_table.php | 2025_03_

database > seeders > LevelSeeder.php

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
    }
}
```

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

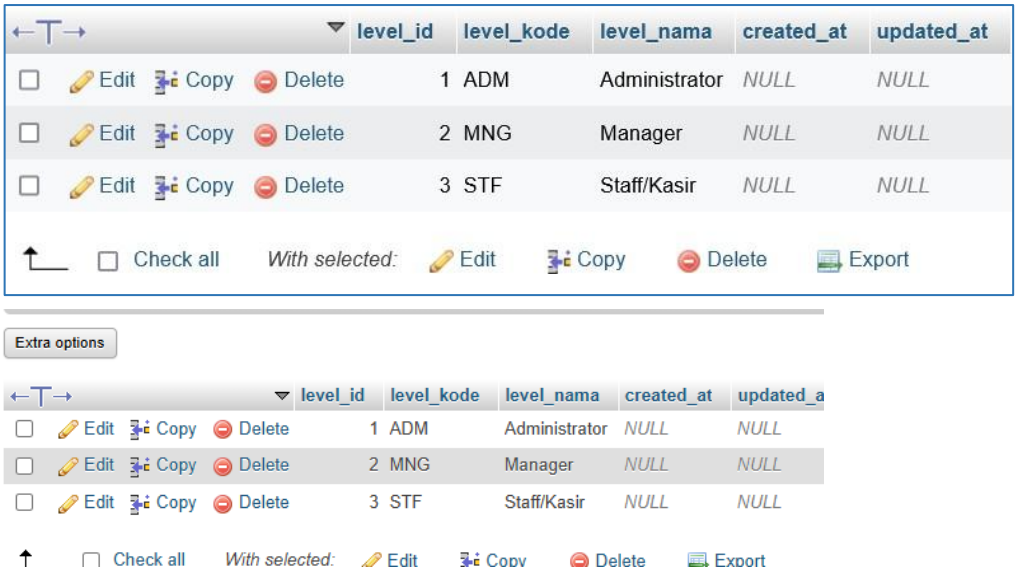


3. Selanjutnya, kita jalankan file *seeder* untuk table `m_level` pada terminal

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

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan db:seed --class=LevelSeeder

  INFO  Seeding database.

PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS>
```

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



	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

Extra options

	level_id	level_kode	level_nama	created_at	updated_a
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

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

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

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:seeder UserSeeder

  INFO  Seeder [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\seeders\UserSeeder.php] cr
eated successfully.

PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS>
```

6. Modifikasi file `class UserSeeder` seperti berikut

```php
class UserSeeder extends Seeder
{
    public function run(): void
    {
        $data = [
            [
                'user_id' => 1,
                'level_id' => 1,
                'username' => 'admin',
                'nama' => 'Administrator',
                'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
            ],
            [
                'user_id' => 2,
                'level_id' => 2,
                'username' => 'manager',
                'nama' => 'Manager',
                'password' => Hash::make('12345'),
            ],
            [
                'user_id' => 3,
                'level_id' => 3,
                'username' => 'staff',
                'nama' => 'Staff/Kasir',
                'password' => Hash::make('12345'),
            ],
        ];
        DB::table('m_user')->insert($data);
    }
}
```

```php
<?php

namespace Database\Seeders;

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

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            [
                'user_id' => 1,
                'level_id' => 1,
                'username' => 'admin',
                'nama' => 'Administrator',
                'password' => Hash::make('12345'),
            ],
            [
                'user_id' => 2,
                'level_id' => 2,
                'username' => 'manager',
                'nama' => 'Manager',
                'password' => Hash::make('12345'),
            ],
            [
                'user_id' => 3,
                'level_id' => 3,
                'username' => 'staff',
                'nama' => 'Staff/Kasir',
                'password' => Hash::make('12345'),
            ],
        ];

        DB::table('m_user')->insert($data);
    }
}
```

7. Jalankan perintah untuk mengeksekusi class `UserSeeder`

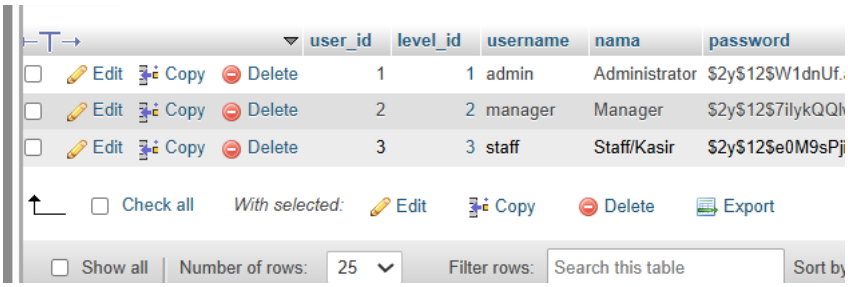
```
php artisan db:seed --class=UserSeeder
```

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan db:seed --class=UserSeeder
  INFO  Seeding database.
```

8. Perhatikan hasil seeder pada table `m_user`

	user_id	level_id	username	nama	password
Edit Copy Delete	1	1	admin	Administrator	$2y$12$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6fXV...
Edit Copy Delete	2	2	manager	Manager	$2y$12$Ajfns20/FdPTeUgghz31muEhlFaruLxkh5wvZ9NGRpu...
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMy/BHnbJ9W...

Check all With selected: Edit Copy Delete Export



	user_id	level_id	username	nama	password
Edit Copy Delete	1	1	admin	Administrator	$2y$12$W1dnUf.
Edit Copy Delete	2	2	manager	Manager	$2y$12$7ilykQQl
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$e0M9sPj

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by

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

A. m_kategori

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:seeder KategoriSeeder
>>
```

database / seeders / KategoriSeeder.php

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class KategoriSeeder extends Seeder
{
    public function run(): void
    {
        $data = [
            ['kategori_id' => 1, 'kategori_kode' => 'ELEC', 'kategori_nama' => 'Elektronik'],
            ['kategori_id' => 2, 'kategori_kode' => 'PAK', 'kategori_nama' => 'Pakaian'],
            ['kategori_id' => 3, 'kategori_kode' => 'MAK', 'kategori_nama' => 'Makanan'],
            ['kategori_id' => 4, 'kategori_kode' => 'MIN', 'kategori_nama' => 'Minuman'],
            ['kategori_id' => 5, 'kategori_kode' => 'ALT', 'kategori_nama' => 'Alat Tulis'],
        ];

        DB::table('m_kategori')->insert($data);
    }
}
```

```
                                                    php artisan db:seed --class=KategoriSeeder
> C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS>

  INFO  Seeding database.
```

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by

Extra options

	kategori_id	kategori_kode	kategori_nama	created_at	up
Edit Copy Delete	1	ELEC	Elektronik	NULL	NU
Edit Copy Delete	2	PAK	Pakaian	NULL	NU
Edit Copy Delete	3	MAK	Makanan	NULL	NU
Edit Copy Delete	4	MIN	Minuman	NULL	NU
Edit Copy Delete	5	ALT	Alat Tulis	NULL	NU

Check all | With selected: Edit Copy Delete Export

B. m_barang

```
INFO  Seeder [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\seeders\BarangSeeder.php] created successfully.
```

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class BarangSeeder extends Seeder
{
    public function run(): void
    {
        $data = [
            ['barang_id' => 1, 'nama_barang' => 'Laptop', 'harga' => 5000000, 'stok' => 10, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 2, 'nama_barang' => 'Smartphone', 'harga' => 3000000, 'stok' => 15, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 3, 'nama_barang' => 'Jaket', 'harga' => 250000, 'stok' => 25, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 4, 'nama_barang' => 'Celana Jeans', 'harga' => 200000, 'stok' => 20, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 5, 'nama_barang' => 'Roti', 'harga' => 15000, 'stok' => 50, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 6, 'nama_barang' => 'Mie Instan', 'harga' => 5000, 'stok' => 100, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 7, 'nama_barang' => 'Teh Botol', 'harga' => 5000, 'stok' => 30, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 8, 'nama_barang' => 'Kopi Sachet', 'harga' => 2000, 'stok' => 60, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 9, 'nama_barang' => 'Buku', 'harga' => 50000, 'stok' => 40, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['barang_id' => 10, 'nama_barang' => 'Pensil', 'harga' => 5000, 'stok' => 75, 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
        ];

        DB::table('m_barang')->insert($data);
    }
}
```

	barang_id	nama_barang	harga	stok	created_at	updated_at
Edit Copy Delete	1	Laptop	5000000	10	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	2	Smartphone	3000000	15	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	3	Jaket	250000	25	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	4	Celana Jeans	200000	20	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	5	Roti	15000	50	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	6	Mie Instan	5000	100	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	7	Teh Botol	5000	30	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	8	Kopi Sachet	2000	60	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	9	Buku	50000	40	2025-03-04 13:26:13	2025-03-04 13:26:13
Edit Copy Delete	10	Pensil	5000	75	2025-03-04 13:26:13	2025-03-04 13:26:13

C. t_stok

```php
namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class StokSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
    }
}
```

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class StokSeeder extends Seeder
{
    public function run(): void
    {
        // Data stok sesuai dengan jumlah barang di m_barang
        $data = [
            ['stok_id' => 1, 'barang_id' => 1, 'jumlah' => 50, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 2, 'barang_id' => 2, 'jumlah' => 30, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 3, 'barang_id' => 3, 'jumlah' => 40, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 4, 'barang_id' => 4, 'jumlah' => 25, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 5, 'barang_id' => 5, 'jumlah' => 100, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 6, 'barang_id' => 6, 'jumlah' => 75, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 7, 'barang_id' => 7, 'jumlah' => 60, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 8, 'barang_id' => 8, 'jumlah' => 90, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 9, 'barang_id' => 9, 'jumlah' => 55, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['stok_id' => 10, 'barang_id' => 10, 'jumlah' => 80, 'keterangan' => 'Stok awal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
        ];

        DB::table('t_stok')->insert($data);
    }
}
```

Extra options

	stok_id	barang_id	jumlah	keterangan	created_at
Edit Copy Delete	1	1	50	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	2	2	30	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	3	3	40	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	4	4	25	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	5	5	100	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	6	6	75	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	7	7	60	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	8	8	90	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	9	9	55	Stok awal	2025-03-04 13:34:18
Edit Copy Delete	10	10	80	Stok awal	2025-03-04 13:34:18

D. t_penjualan

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:seeder PenjualanSeeder
>>

  INFO  Seeder [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\seeders\PenjualanSeeder.php] created successfully.
```

```php
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class PenjualanSeeder extends Seeder
{
    public function run(): void
    {
        // Data transaksi penjualan
        $data = [
            ['penjualan_id' => 1, 'tanggal' => '2024-03-01', 'nama_pelanggan' => 'Andi', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 2, 'tanggal' => '2024-03-02', 'nama_pelanggan' => 'Budi', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 3, 'tanggal' => '2024-03-03', 'nama_pelanggan' => 'Citra', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 4, 'tanggal' => '2024-03-04', 'nama_pelanggan' => 'Dewi', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 5, 'tanggal' => '2024-03-05', 'nama_pelanggan' => 'Eko', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 6, 'tanggal' => '2024-03-06', 'nama_pelanggan' => 'Faisal', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 7, 'tanggal' => '2024-03-07', 'nama_pelanggan' => 'Gita', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 8, 'tanggal' => '2024-03-08', 'nama_pelanggan' => 'Hendra', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 9, 'tanggal' => '2024-03-09', 'nama_pelanggan' => 'Indah', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
            ['penjualan_id' => 10, 'tanggal' => '2024-03-10', 'nama_pelanggan' => 'Joko', 'created_at' => Carbon::now(), 'updated_at' => Carbon::now()],
        ];

        DB::table('t_penjualan')->insert($data);
    }
}
```

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan db:seed --class=PenjualanSeeder
>>

  INFO  Seeding database.
```

penjualan_id	tanggal	nama_pelanggan	created_at	updated_at
1	2024-03-01	Andi	2025-03-04 13:39:01	2025-03-04 13:39:01
2	2024-03-02	Budi	2025-03-04 13:39:01	2025-03-04 13:39:01
3	2024-03-03	Citra	2025-03-04 13:39:01	2025-03-04 13:39:01
4	2024-03-04	Dewi	2025-03-04 13:39:01	2025-03-04 13:39:01
5	2024-03-05	Eko	2025-03-04 13:39:01	2025-03-04 13:39:01
6	2024-03-06	Faisal	2025-03-04 13:39:01	2025-03-04 13:39:01
7	2024-03-07	Gita	2025-03-04 13:39:01	2025-03-04 13:39:01
8	2024-03-08	Hendra	2025-03-04 13:39:01	2025-03-04 13:39:01
9	2024-03-09	Indah	2025-03-04 13:39:01	2025-03-04 13:39:01
10	2024-03-10	Joko	2025-03-04 13:39:01	2025-03-04 13:39:01

E. t_penjualan_detail

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:seeder PenjualanDetailSeeder

  INFO  Seeder [C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS\database\seeders\PenjualanDetailSeeder.php] created successfully.
```

	id	penjualan_id	barang_id	jumlah	harga	created_at	updated_at
Edit Copy Delete	1	1	1	2	2000000.00	2025-03-04 14:18:12	2025-03-04 14:
Edit Copy Delete	2	1	2	1	500000.00	2025-03-04 14:18:12	2025-03-04 14:
Edit Copy Delete	3	2	3	3	1500000.00	2025-03-04 14:18:12	2025-03-04 14:18:12

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;

class PenjualanDetailSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        DB::table('t_penjualan_detail')->insert([
            [
                'penjualan_id' => 1,
                'barang_id'    => 1,
                'jumlah'       => 2,
                'harga'        => 2000000,
                'created_at'   => Carbon::now(),
                'updated_at'   => Carbon::now(),
            ],
            [
                'penjualan_id' => 1,
                'barang_id'    => 2,
                'jumlah'       => 1,
                'harga'        => 500000,
                'created_at'   => Carbon::now(),
                'updated_at'   => Carbon::now(),
            ],
            [
                'penjualan_id' => 2,
                'barang_id'    => 3,
                'jumlah'       => 3,
                'harga'        => 1500000,
                'created_at'   => Carbon::now(),
                'updated_at'   => Carbon::now(),
            ],
        ]);
    }
}
```

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 perinta 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
```

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:controller LevelController
```

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

LevelController.php | web.php

routes > web.php > ...

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

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

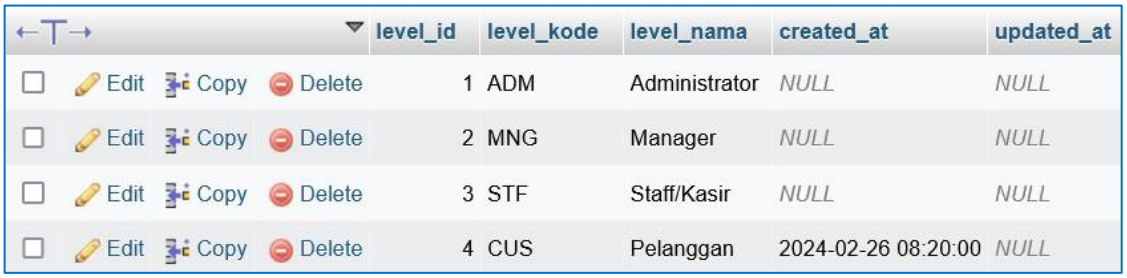
LevelController.php × web.php

app > Http > Controllers > LevelController.php > ...

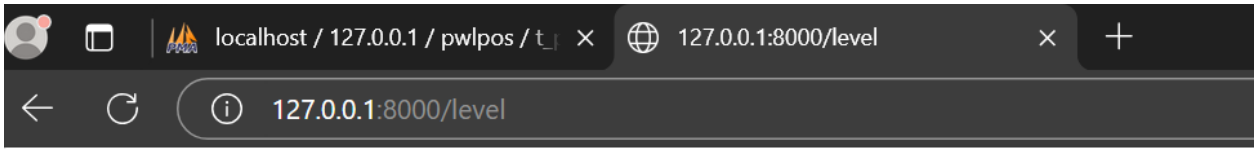
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController 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';
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index()
    {
        DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', 'Pela

        return 'Insert data baru berhasil';
    }
}
```

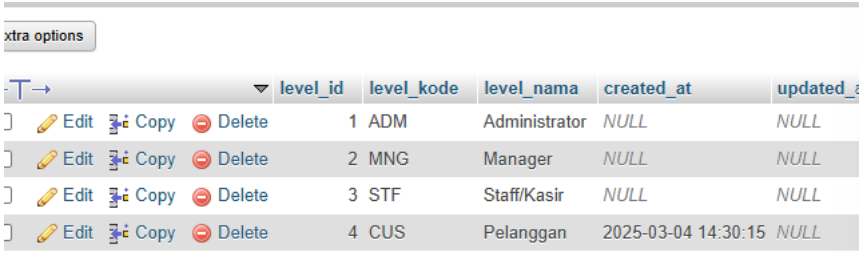
4. Kita coba jalankan di browser dengan url `localhost/PWL_POS/public/level` dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`



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



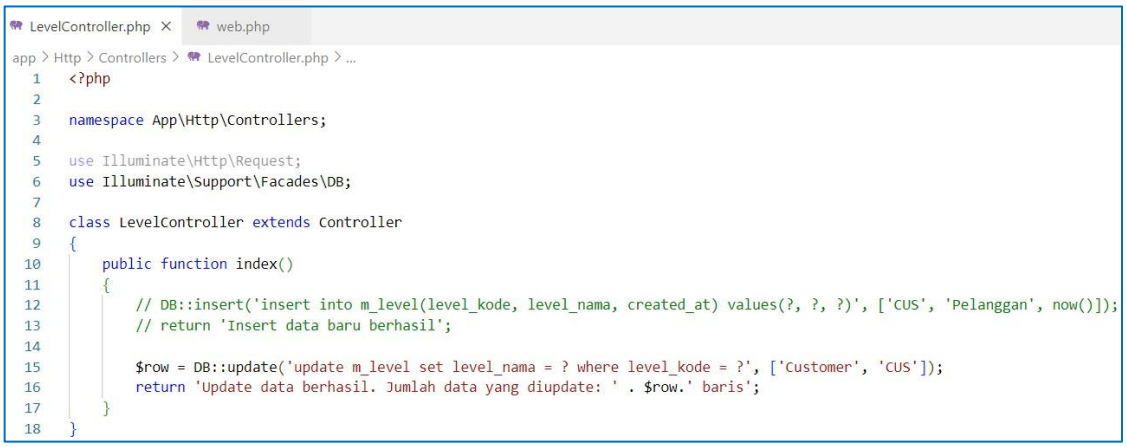
Insert data baru berhasil



xtra options

	level_id	level_kode	level_nama	created_at	updated_a
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
Edit Copy Delete	4	CUS	Pelanggan	2025-03-04 14:30:15	NULL

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



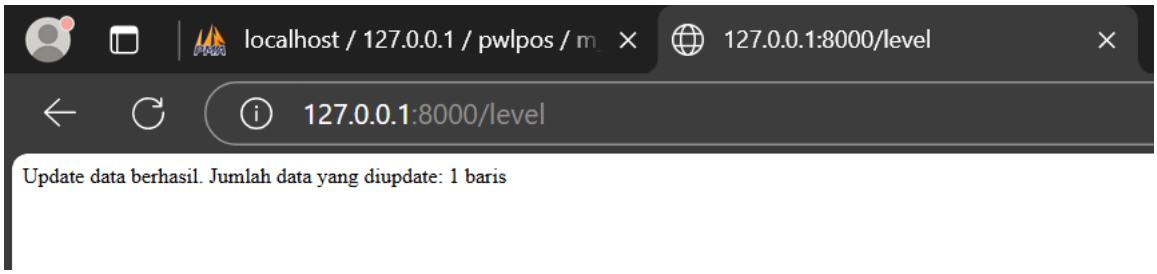
LevelController.php × web.php

app > Http > Controllers > LevelController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController 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';

        $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';
    }
}
```

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`

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', 'P
        // return 'Insert data baru berhasil';

        $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
    }
}
```



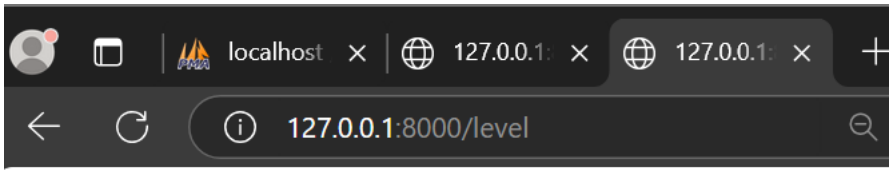
7. Kita coba modifikasi lagi file `LevelController` untuk melakukan proses hapus data

LevelController.php | web.php

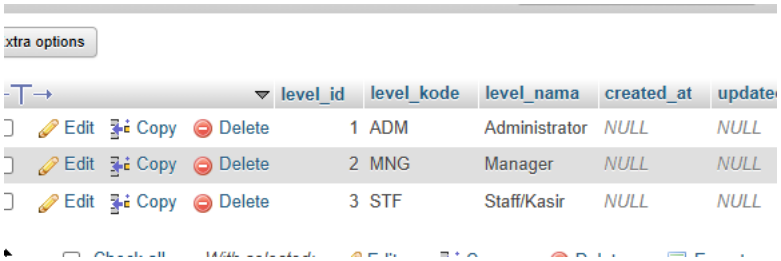
app > Http > Controllers > LevelController.php > LevelController > index

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController 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';

        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';

        $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
        return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row.' baris';
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS'
        // return 'Insert data baru berhasil';

        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS'
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';

        $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
        return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
    }
}
```



Delete data berhasil. Jumlah data yang dihapus: 1 baris



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

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController 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';

        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';

        // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
        // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row.' baris';

        $data = DB::select('select * from m_level');
        return view('level', ['data' => $data]);
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', 'P
        // return 'Insert data baru berhasil';

        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';

        // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
        // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';

        $data = DB::select('select * from m_level');
        return view('level', ['data' => $data]);
    }
}

```

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`

LevelController.php | level.blade.php | web.php

resources > views > level.blade.php > ...

```blade
<!DOCTYPE html>
<html>
    <head>
        <title>Data Level Pengguna</title>
    </head>
    <body>
        <h1>Data Level Pengguna</h1>
        <table border="1" cellpadding="2" cellspacing="0">
            <tr>
                <th>ID</th>
                <th>Kode Level</th>
                <th>Nama Level</th>
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->level_id }}</td>
                <td>{{ $d->level_kode }}</td>
                <td>{{ $d->level_nama }}</td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

Data Level Pengguna

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

10. Silahkan dicoba pada browser dan amati apa yang terjadi
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

a. 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;`

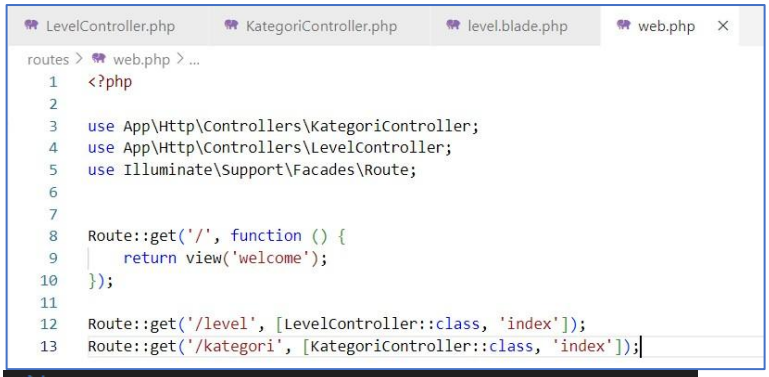
Praktikum 5 – Implementasi *Query Builder*

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

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

```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:controller KategoriController
```

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



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

LevelController.php | KategoriController.php | level.blade.php | web.php

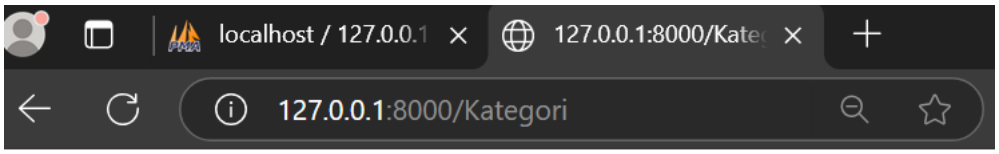
app > Http > Controllers > KategoriController.php > KategoriController > index

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class KategoriController extends Controller
{
    public function index()
    {
        $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil';
    }
}
```

app > Http > Controllers > KategoriController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class KategoriController extends Controller
{
    public function index()
    {
        $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];

        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil';
    }
}
```

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`



nsert data baru berhasil

	kategori_id	kategori_kode	kategori_nama	created_at	up
Edit Copy Delete	1	ELEC	Elektronik	NULL	NU
Edit Copy Delete	2	PAK	Pakaian	NULL	NU
Edit Copy Delete	3	MAK	Makanan	NULL	NU
Edit Copy Delete	4	MIN	Minuman	NULL	NU
Edit Copy Delete	5	ALT	Alat Tulis	NULL	NU
Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2025-03-04 14:53:45	NU

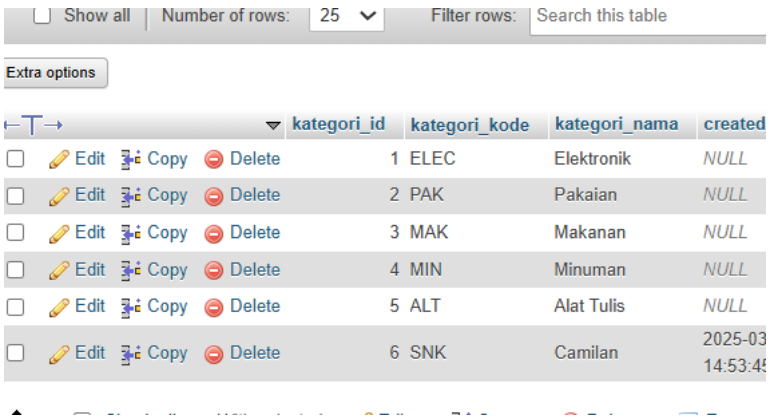
5. Selanjutnya, kita modifikasi lagi file `KategoriController` untuk meng-*update* data di table `m_kategori` seperti berikut

app > Http > Controllers > KategoriController.php > KategoriController > index

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class KategoriController extends Controller
{
    public function index()
    {
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

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

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class KategoriController extends Controller
{
    public function index()
    {
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];

        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

        $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan'])

        return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
    }
}
```

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`



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

```php
    public function index()
    {
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

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

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

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class KategoriController extends Controller
{
    public function index()
    {
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];

        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

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

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

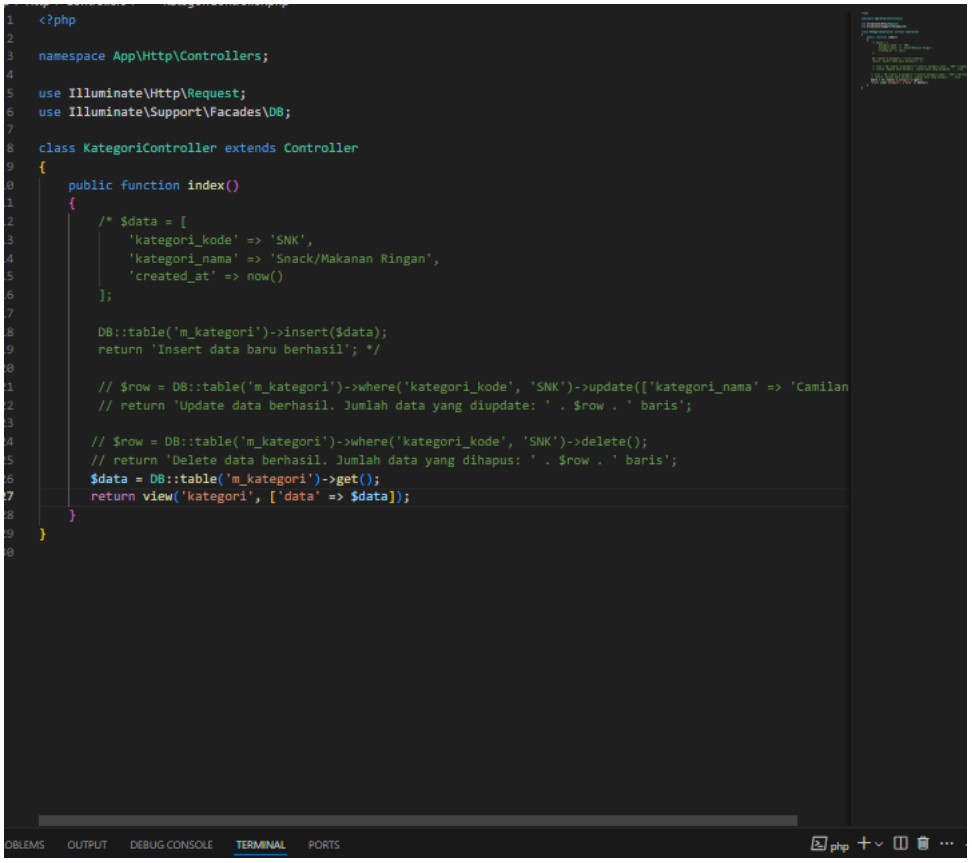
Extra options

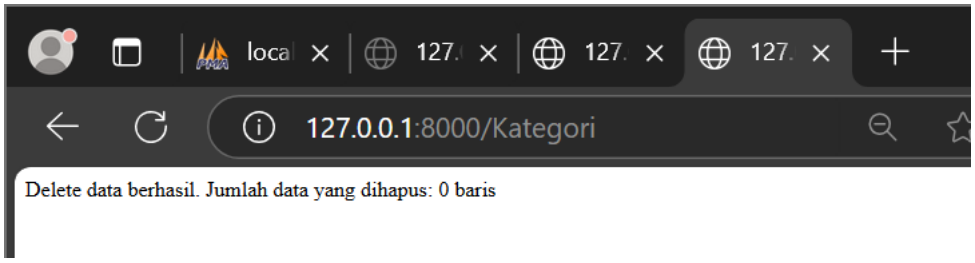
	kategori_id	kategori_kode	kategori_nama	created_at
Edit Copy Delete	1	ELEC	Elektronik	NULL
Edit Copy Delete	2	PAK	Pakaian	NULL
Edit Copy Delete	3	MAK	Makanan	NULL
Edit Copy Delete	4	MIN	Minuman	NULL
Edit Copy Delete	5	ALT	Alat Tulis	NULL

Check all With selected: Edit Copy Delete Export

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

```php
    public function index()
    {
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

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

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

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





	kategori_id	kategori_kode	kategori_nama
Edit Copy Delete	1	ELEC	Elektronik
Edit Copy Delete	2	PAK	Pakaian
Edit Copy Delete	3	MAK	Makanan
Edit Copy Delete	4	MIN	Minuman
Edit Copy Delete	5	ALT	Alat Tulis

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

```
<!DOCTYPE html>
<html>
    <head>
        <title>Data Kategori Barang</title>
    </head>
    <body>
        <h1>Data Kategori Barang</h1>
        <table border="1" cellpadding="2" cellspacing="0">
            <tr>
                <th>ID</th>
                <th>Kode Kategori</th>
                <th>Nama Kategori</th>
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->kategori_id }}</td>
                <td>{{ $d->kategori_kode }}</td>
                <td>{{ $d->kategori_nama }}</td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	ELEC	Elektronik
2	PAK	Pakaian
3	MAK	Makanan
4	MIN	Minuman
5	ALT	Alat Tulis

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

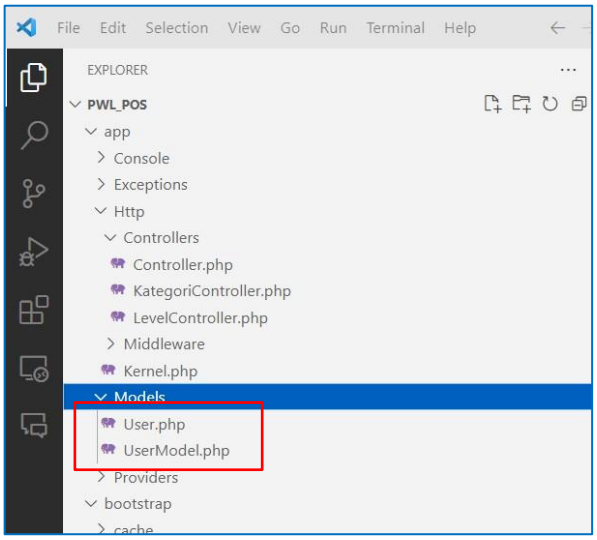
Untuk bisa melakukan operasi ***CRUD*** (*create, read/retrieve, update, delete*), kita harus membuat sebuah model sesuai dengan target tabel yang ingin digunakan. Jadi,

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
```

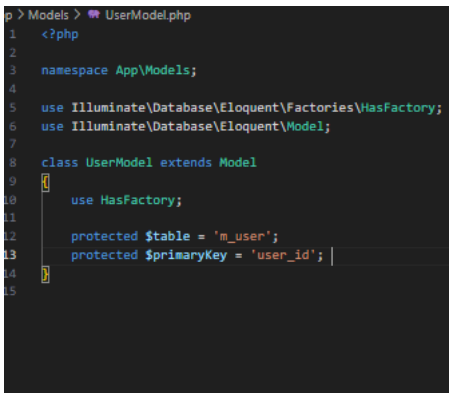


```
PS C:\Users\RYZEN\PemrogramanWebLanjut_2025\PWL2025_Jobsheet3\PWL_POS> php artisan make:model UserModel
```

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

app > Models > UserModel.php > UserModel

```php
<?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
<?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';
    protected $primaryKey = 'user_id';
}
```

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

routes > web.php > ...

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;


Route::get('/', function () {
    return view('welcome');
});

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

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

app > Http > Controllers > UserController.php > ...

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index()
    {
        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view('user', ['data' => $user]);
    }
}
```

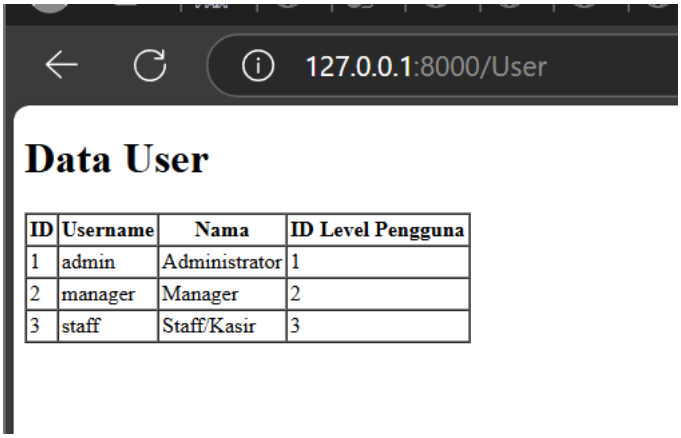
6. Kemudian kita buat view `user.blade.php`

resources > views > user.blade.php > ...

```
<!DOCTYPE html>
<html>
    <head>
        <title>Data User</title>
    </head>
    <body>
        <h1>Data User</h1>
        <table border="1" cellpadding="2" cellspacing="0">
            <tr>
                <th>ID</th>
                <th>Username</th>
                <th>Nama</th>
                <th>ID Level Pengguna</th>
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->user_id }}</td>
                <td>{{ $d->username }}</td>
                <td>{{ $d->nama }}</td>
                <td>{{ $d->level_id }}</td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Data User</title>
</head>
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>ID</th>
            <th>Username</th>
            <th>Nama</th>
            <th>ID Level Pengguna</th>
        </tr>
        @foreach ($data as $d)
        <tr>
            <td>{{ $d->user_id }}</td>
            <td>{{ $d->username }}</td>
            <td>{{ $d->nama }}</td>
            <td>{{ $d->level_id }}</td>
        </tr>
        @endforeach
    </table>
</body>
</html>

```

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



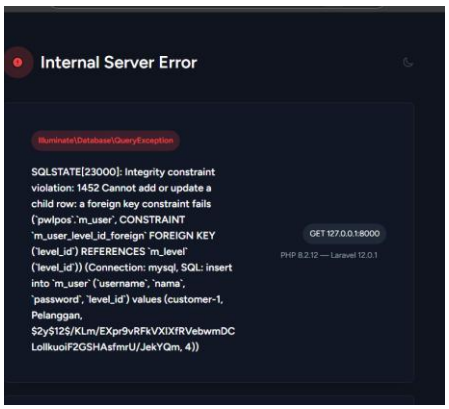
8. Setelah itu, kita modifikasi lagi file `UserController`

app > Http > Controllers > UserController.php > ...

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        // tambah data user dengan Eloquent Model
        $data = [
            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make('12345'),
            'level_id' => 4
        ];
        UserModel::insert($data); // tambahkan data ke tabel m_user

        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view('user', ['data' => $user]);
    }
}
```

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {

        $data = [
            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make('12345'),
            'level_id' => 4
        ];

        UserModel::insert($data);


        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

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



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

```php
class UserController extends Controller
{
    public function index()
    {
        // tambah data user dengan Eloquent Model
        $data = [
            'nama' => 'Pelanggan Pertama',
        ];
        UserModel::where('username', 'customer-1')->update($data); // update data user

        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view('user', ['data' => $user]);
    }
}
```

app > Http > Controllers > UserController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index()
    {

        $data = [
            'nama' => 'Pelanggan Pertama',
        ];

        UserModel::where('username', 'customer-1')->update($data); // update data user


        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}

```

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?
2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk `APP_KEY`?
3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/*output* dari fungsi tersebut?
5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?
7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
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')` ?
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (`?`), apa kegunaan dari tanda tanya (`?`) tersebut?
11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table = ‘m_user’;` dan `protected $primaryKey = ‘user_id’;` ?
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade* / *Query Builder* / *Eloquent ORM*) ? jelaskan

Berikut jawaban atas pertanyaan penutup berdasarkan pemahaman materi di atas:

1. Fungsi dari APP_KEY pada file setting .env Laravel
   APP_KEY digunakan untuk enkripsi data dalam Laravel.

2. Cara men-generate nilai untuk APP_KEY
   php artisan key:generate

3. Jumlah file migrasi default Laravel dan fungsinya
   Secara default, Laravel memiliki beberapa file migrasi, di antaranya:
   - create_users_table berfungsi Membuat tabel pengguna.
   - create_password_resets_table berfungsi Menyimpan token reset password.
   - create_failed_jobs_table berfungsi Menyimpan informasi pekerjaan (jobs) yang gagal dieksekusi.
   - create_personal_access_tokens_table berfungsi untuk API token Laravel.

4.Tujuan dari `$table->timestamps();` dalam file migrasi
   Kode ini secara otomatis menambahkan dua kolom dalam tabel database:
   - created_at berfungsi Menyimpan waktu pembuatan data.
   - updated_at berfungsi Menyimpan waktu terakhir data diperbarui.

5.Tipe data yang dihasilkan dari `$table->id();`
   Fungsi `$table->id();` menghasilkan kolom primary key secara default.

6. Perbedaan `$table->id();` dan `$table->id('level_id');`
   - `$table->id();` Secara default menghasilkan kolom `id` sebagai primary key.
   - `$table->id('level_id');` Menghasilkan kolom `level_id` sebagai primary key, bukan `id`.

7. Fungsi `->unique()` dalam migration
   `->unique()` digunakan untuk memastikan bahwa nilai dalam suatu kolom bersifat unik, tidak ada duplikasi data.

8. Perbedaan `$table->unsignedBigInteger('level_id')` dan `$table->id('level_id')`
   - `$table->id('level_id')` digunakan untuk membuat primary key dengan tipe unsignedBigInteger
   - `$table->unsignedBigInteger('level_id')` digunakan untuk membuat kolom referensi ke primary key tabel lain (foreign key) tanpa menjadikannya primary key.

9. Tujuan dari `Class Hash` dan `Hash::make('1234');`
   - `Hash` digunakan untuk melakukan hashing data, terutama password.
   - `Hash::make('1234');` menghasilkan string hash dari password `1234` agar tidak tersimpan dalam bentuk plaintext di database.

10. Kegunaan tanda tanya `?` pada Query Builder
    Tanda `?` digunakan sebagai **parameter binding** untuk mencegah SQL Injection. Nilai akan diisi secara dinamis saat query dijalankan.

11. Tujuan penulisan `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` dalam Model
    - `protected $table = 'm_user';` → Menentukan bahwa model ini mewakili tabel `m_user`.
    - `protected $primaryKey = 'user_id';` → Menentukan bahwa primary key dari tabel `m_user` adalah `user_id`, bukan `id`.

12. Perbandingan DB Façade, Query Builder, dan Eloquent ORM dalam operasi CRUD*
    - DB Façade: Paling mirip dengan raw SQL query. Cocok untuk query kompleks dan performa tinggi tetapi kurang fleksibel.
    - Query Builder: Mempermudah query dengan sintaks method chaining. Tidak sekompleks ORM tetapi lebih fleksibel dari DB Façade.
    - Eloquent ORM: Paling mudah digunakan karena berbasis objek, tetapi bisa sedikit lebih lambat dibandingkan Query Builder.

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