



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)
Nama : Dea Putri Nastiti
NIM : 2241720117
Kelas : TI 2H

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**.

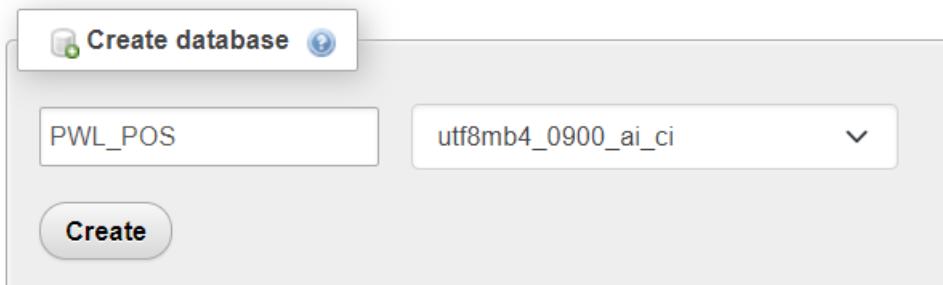
Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari



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:

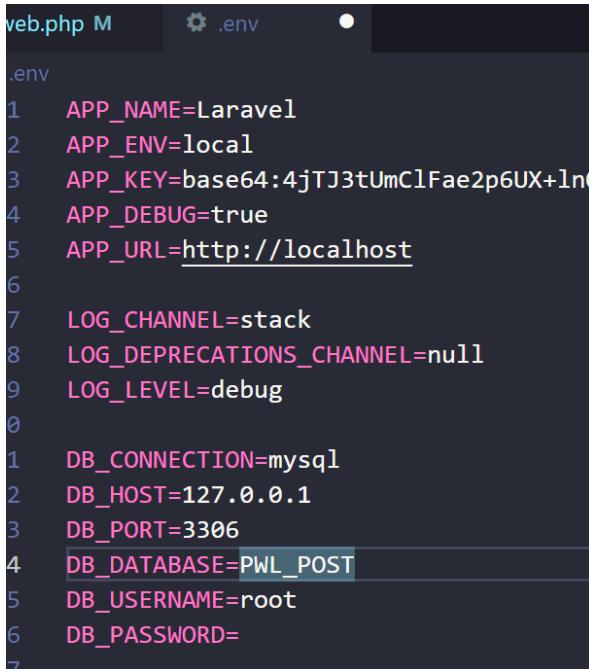


2. Buka aplikasi VSCode dan buka folder project **PWL_POS** yang sudah kita buat
3. Copy file **.env.example** menjadi **.env**
4. Buka file **.env**, dan pastikan konfigurasi **APP_KEY** bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan **php artisan**.

```
⚙️ .env      X   $ .env.example
⚙️ .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:4jTJ3tUmClFae2p6UX+lnOCsPDk9uOEe+i1N2xsU4XI=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
```

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

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



```
web.php M .env ●  
.env  
1 APP_NAME=Laravel  
2 APP_ENV=local  
3 APP_KEY=base64:4jTJ3tUmClFae2p6UX+lnO  
4 APP_DEBUG=true  
5 APP_URL=http://localhost  
6  
7 LOG_CHANNEL=stack  
8 LOG_DEPRECATIONS_CHANNEL=null  
9 LOG_LEVEL=debug  
0  
1 DB_CONNECTION=mysql  
2 DB_HOST=127.0.0.1  
3 DB_PORT=3306  
4 DB_DATABASE=PWL_POST  
5 DB_USERNAME=root  
6 DB_PASSWORD=  
7
```

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

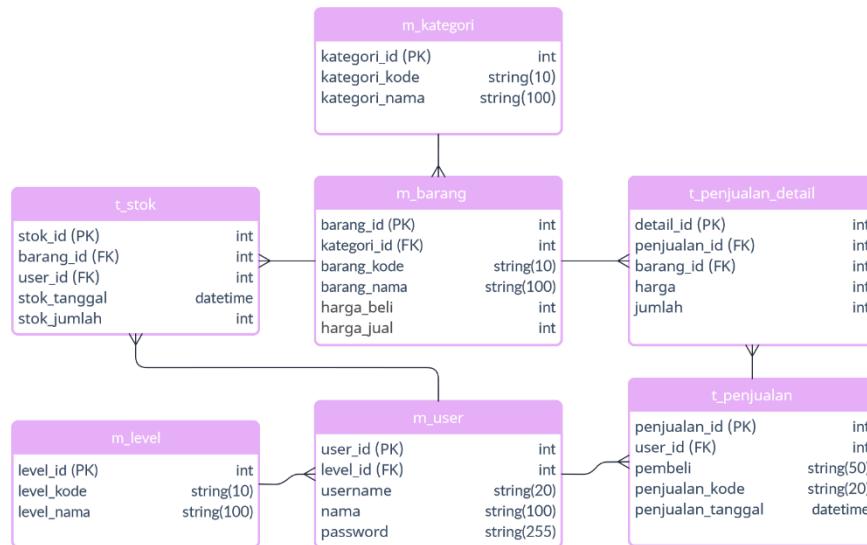
B. MIGRATION

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

- Menggunakan `artisan` untuk membuat *file migration*

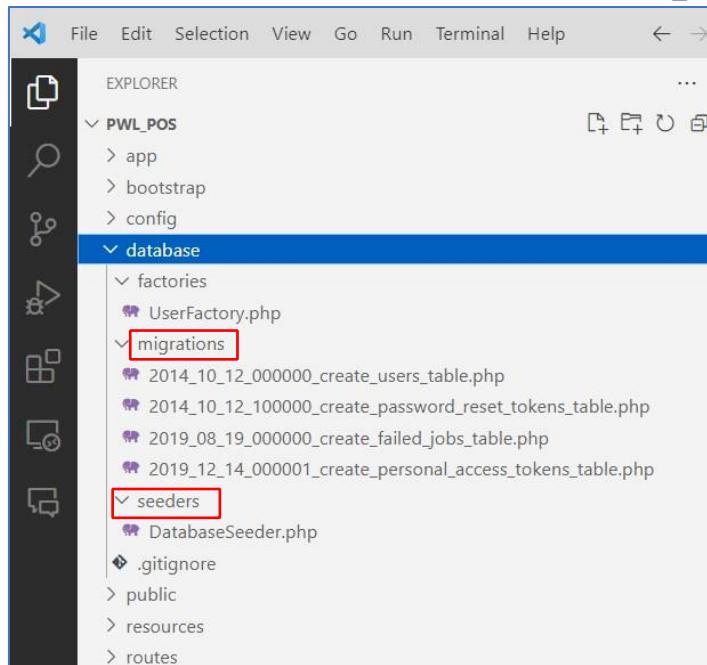
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- Menggunakan `artisan` untuk membuat *file model + file migration*

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

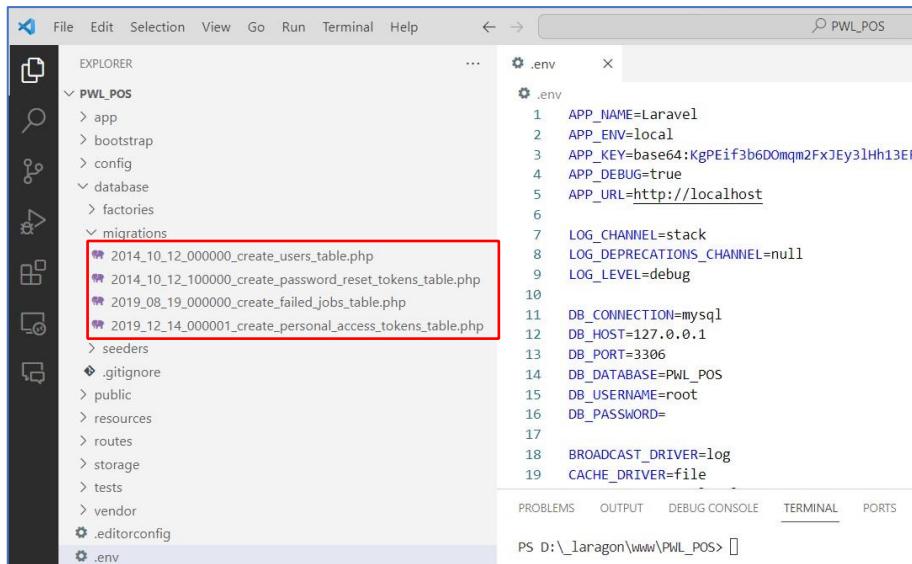
Perintah `-m` di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

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



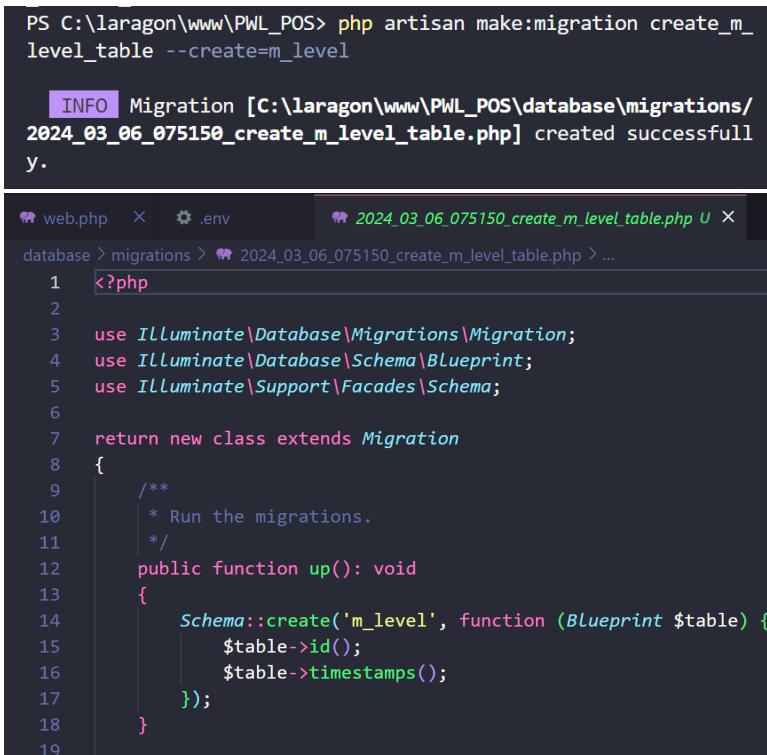
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of the project structure. Under the 'PWL_POS' folder, there are several sub-folders like 'app', 'bootstrap', 'config', 'database', 'factories', 'migrations', 'seeders', '.gitignore', 'public', 'resources', 'routes', 'storage', 'tests', 'vendor', '.editorconfig', and '.env'. The 'migrations' folder contains four files: '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', and '2019_12_14_000001_create_personal_access_tokens_table.php'. The file '2019_12_14_000001_create_personal_access_tokens_table.php' is highlighted with a red box. On the right is the code editor showing the contents of the '.env' file. Below the code editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The terminal tab shows the command '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



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

[INFO] Migration [C:\laragon\www\PWL_POS\database\migrations\2024_03_06_075150_create_m_level_table.php] created successfully.

web.php X .env 2024_03_06_075150_create_m_level_table.php U X

database > migrations > 2024_03_06_075150_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 }
```

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  };
```

```
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
```

INFO

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

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

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



```
php artisan migrate
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

INFO | Preparing database.

Creating migration table 12ms DONE

INFO | Running migrations.

2014_10_12_000000_create_users_table 16ms DONE

2014_10_12_100000_create_password_reset_tokens_table 6ms DONE

2019_08_19_000000_create_failed_jobs_table 42ms DONE

2019_12_14_000001_create_personal_access_tokens_table 15ms DONE

2024_02_25_133526_create_m_level_table 13ms DONE

PS D:\laragon\www\PWL_POS>

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

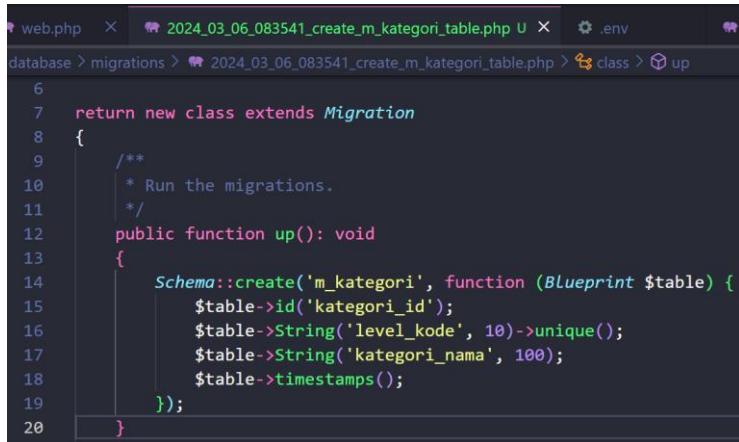
The screenshot shows the phpMyAdmin interface. On the left, there is a tree view of databases and tables under 'pw1_pos'. The 'm_level' table is highlighted with a red box. On the right, there is a grid view of the 'm_level' table with columns: level_id, level_kode, level_nama, created_at, and updated_at. The grid shows one row of data.

Table	Action	Rows
failed_jobs	Browse Structure Insert Empty Drop	0
migrations	Browse Structure Insert Empty Drop	5
m_level	Browse Structure Insert Empty Drop	0
password_reset_tokens	Browse Structure Insert Empty Drop	0
personal_access_tokens	Browse Structure Insert Empty Drop	0
users	Browse Structure Insert Empty Drop	0

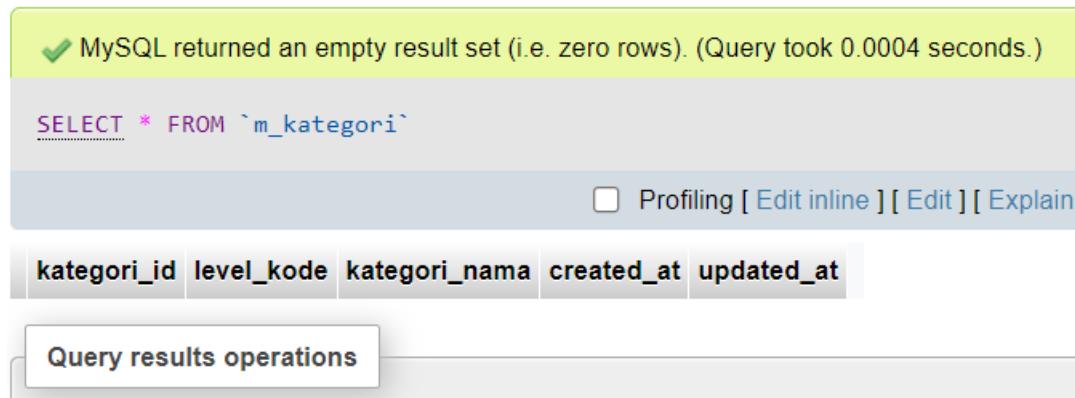
7. Ok, table sudah dibuat di database
8. Buat table *database* dengan *migration* untuk table **m_kategori** yang sama-sama tidak memiliki *foreign key*

```
php artisan make:migration create_m_kategori_table --create=m_kategori
```

INFO Migration [C:\laragon\www\PWL_POS\database\migrations\2024_03_06_083541_create_m_kategori_table.php] created successfully.



```
web.php  X  2024_03_06_083541_create_m_kategori_table.php U  .env
database > migrations > 2024_03_06_083541_create_m_kategori_table.php > class > up
6
7     return new class extends Migration
8     {
9         /**
10          * Run the migrations.
11          */
12         public function up(): void
13         {
14             Schema::create('m_kategori', function (Blueprint $table) {
15                 $table->id('kategori_id');
16                 $table->String('level_kode', 10)->unique();
17                 $table->String('kategori_nama', 100);
18                 $table->timestamps();
19             });
20         }
21     }
```



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0004 seconds.)

```
SELECT * FROM `m_kategori`
```

Profiling [Edit inline] [Edit] [Explain]

kategori_id	level_kode	kategori_nama	created_at	updated_at
-------------	------------	---------------	------------	------------

Query results operations

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

--create=m_user



Praktikum 2.³ - 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:\laragon\www\PWL_POS> php artisan make:migration cr
eate_m_user_table --table=m_user

[INFO] Migration [C:\laragon\www\PWL_POS\database\migr
ations/2024_03_06_085249_create_m_user_table.php] cre
ated successfully.
```

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

```
database > migrations > 2024_03_06_085249_create_m_user_table.php > class > up
  7  return new class extends Migration
  8  {
  9      /**
 10      * Perform up migration.
 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 sam
 18              $table->String('nama', 100);
 19              $table->String('password');
 20              $table->timestamps();
 21
 22              //Mendefinisikan Foreign Key pada kolom level_id di tabel m_level
 23              $table->Foreign('level_id')->references('level_id')->on('m_level');
 24              //
 25          });
 26      }
 27  }
```

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

```
SELECT * FROM `m_user`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

user_id	level_id	username	nama	password	created_at	updated_at
---------	----------	----------	------	----------	------------	------------



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:\laragon\www\PWL_POS> php artisan make:migration c
reate_m_barang_table --create=m
artisan make:migration c
artisan make:migration create_m_barang_table --create=m
_barang

[INFO] Migration [C:\laragon\www\PWL_POS\database\mig
rations/2024_03_06_090751_create_m_barang_table.php] cr
eated successfully.
```

```
database > migrations > 2024_03_06_090751_create_m_barang_table.php > class > up > Closure
Search (Ctrl+Shift+F) new class extends Migration
12     public function up(): void
13     {
14         Schema::create('m_barang', function (Blueprint $table) {
15             $table->id('barang_id');
16             $table->unsignedBigInteger('kategori_id')->index(); //indexing untuk ForeignKey
17             $table->String('barang_kode', 10)->unique();
18             $table->String('barang_nama', 100);
19             $table->integer('harga_beli');
20             $table->integer('harga_jual');
21             $table->timestamps();
22
23             //mendefinisikan Foreign Key pada kolom kategori_id mengacu pada kolom kategori_id di tabel m_kategori
24             $table->foreign('kategori_id')->references('kategori_id')->on('m_kategori');
25         });
26     }
27 }
```

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

[INFO] Running migrations.

2024_03_06_090751_create_m_barang_table ..... 364ms DONE
```

```
SELECT * FROM `m_barang`

 Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP]

```

barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
-----------	-------------	-------------	-------------	------------	------------	------------	------------

[t_penjualan](#)

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_t_penjualan_tabl
e --create=t_penjualan

[INFO] Migration [C:\laragon\www\PWL_POS\database\migrations/2024_03_09_053
754_create_t_penjualan_table.php] created successfully.
```



```
database > migrations > 2024_03_09_053754_create_t_penjualan_table.php > class > up > Closure
  7  return new class extends Migration
  12    public function up(): void
  13    {
  14      Schema::create('t_penjualan', function (Blueprint $table) {
  15        $table->id('penjualan_id');
  16        $table->unsignedBigInteger('user_id')->index(); //Foreign Key
  17        $table->string('pembeli', 50);
  18        $table->string('penjualan_kode', 20)->unique();
  19        $table->dateTime('penjualan_tanggal');
  20        $table->timestamps();
  21
  22        //mendefinisikan Foreign Key pada kolom user_id mengacu pada kolom user_id di tabel m_user
  23        $table->foreign('user_id')->references('user_id')->on('m_user');
  24      });
  25    }
  26  }
```

```
PS C:\laragon\www\PWL_POS> php artisan migrate
INFO Running migrations.
2024_03_09_053754_create_t_penjualan_table ..... 265ms DONE
```

```
SELECT * FROM `t_penjualan`
Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create ]
```

t_stok

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_t_stok_table --create=t_stok
INFO Migration [C:\laragon\www\PWL_POS\database\migrations\2024_03_09_054651_create_t_stok_table.php] created successfully.
```

```
database > migrations > 2024_03_09_054651_create_t_stok_table.php > class > up
  7  return new class extends Migration
  12    public function up(): void
  13    {
  14      Schema::create('t_stok', function (Blueprint $table) {
  15        $table->id('stok_id');
  16        $table->unsignedBigInteger('barang_id')->index(); //Foreign key
  17        $table->unsignedBigInteger('user_id')->index(); //Foreign key
  18        $table->dateTime('stok_tanggal');
  19        $table->integer('stok_jumlah');
  20        $table->timestamps();
  21
  22        //mendefinisikan Foreign Key pada kolom barang_id mengacu pada kolom barang_id di tabel m_barang
  23        $table->foreign('barang_id')->references('barang_id')->on('m_barang');
  24        //mendefinisikan Foreign Key pada kolom user_id mengacu pada kolom user_id di tabel m_user
  25        $table->foreign('user_id')->references('user_id')->on('m_user');
  26      });
  27    }
  28  }
```

```
php artisan migrate
INFO Running migrations.
2024_03_09_054651_create_t_stok_table ..... 442ms DONE
```

t_penjualan_detail

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_t_penjualan_detail_table --create=t_penjualan_detail
INFO Migration [C:\laragon\www\PWL_POS\database\migrations\2024_03_09_054811_create_t_penjualan_detail_table.php] created successfully.
```



```
database > migrations > 2024_03_09_060454_create_t_penjualan_detail_table.php > class > up > Closure
  7   return new class extends Migration
  8   {
  9     public function up(): void
 10    {
 11      Schema::create('t_penjualan_detail', function (Blueprint $table) {
 12        $table->id('detail_id');
 13        $table->unsignedBigInteger('penjualan_id')->index(); //foreign key
 14        $table->unsignedBigInteger('barang_id')->index(); //foreign key
 15        $table->integer('harga');
 16        $table->integer('jumlah');
 17        $table->timestamps();
 18
 19        //mendefinisikan Foreign Key pada kolom penjualan_id mengacu pada kolom penjualan_id di tabel t_penjualan
 20        $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan');
 21        //mendefinisikan Foreign Key pada kolom barang_id mengacu pada kolom barang_id di tabel m_barang
 22        $table->foreign('barang_id')->references('barang_id')->on('m_barang');
 23      });
 24    }
 25  }
 26 }
```

```
PS C:\laragon\www\PWL_POS> php artisan migrate
INFO  Running migrations.

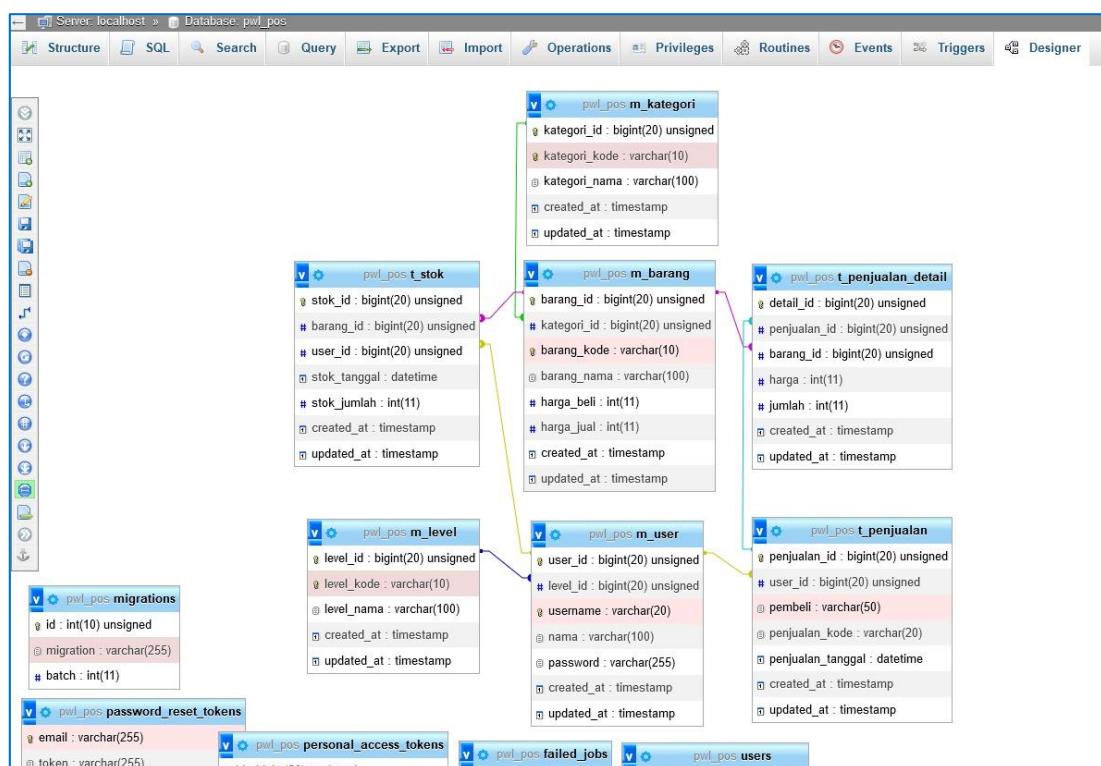
2024_03_09_060454_create_t_penjualan_detail_table ..... 328ms DONE
```

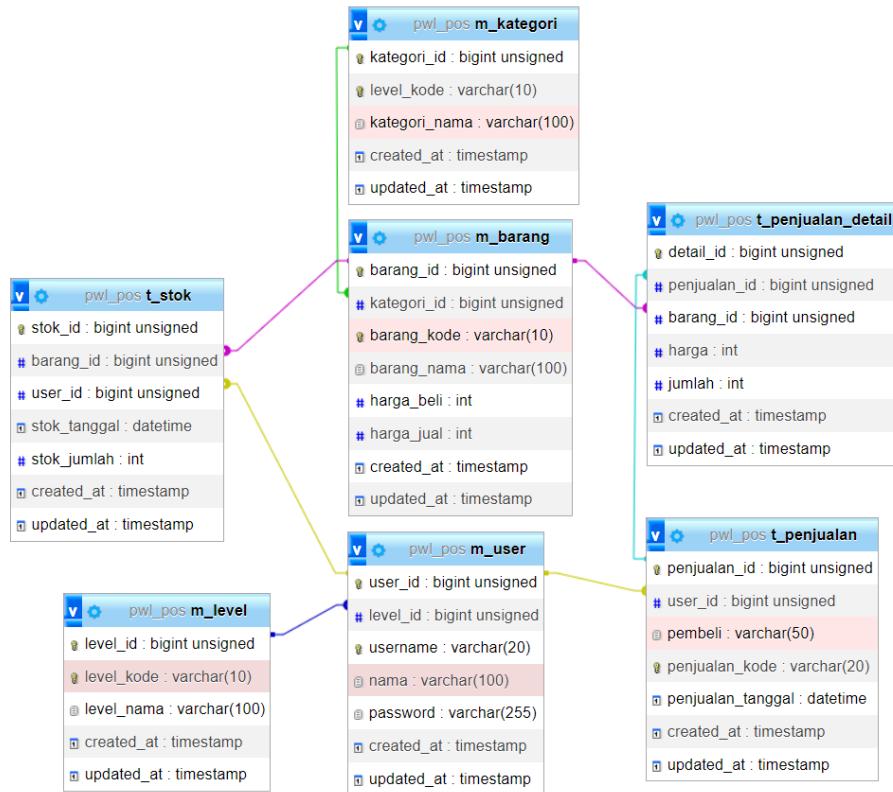
SELECT * FROM `t_penjualan_detail`

Profiling [Edit inline] [Edit] [E]

detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
-----------	--------------	-----------	-------	--------	------------	------------

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>
```



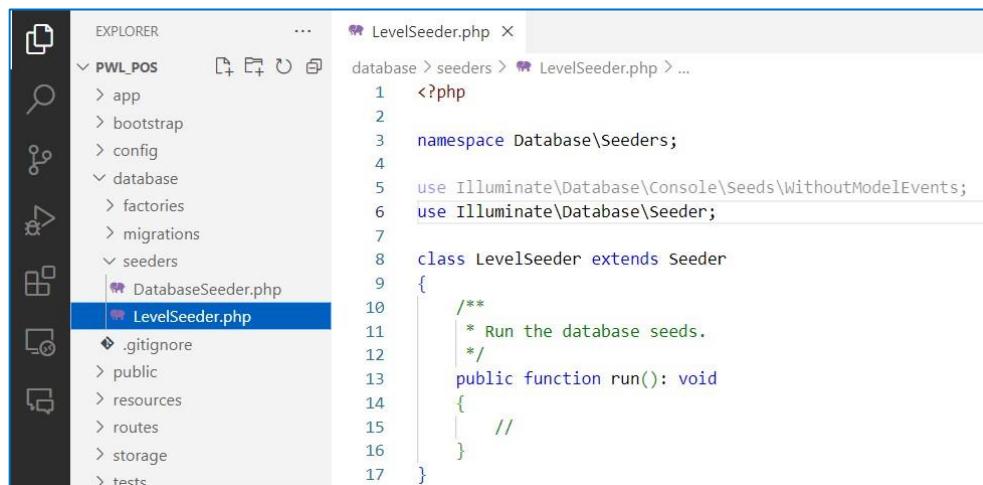
Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau *dummy* data untuk memudahkan pengujian dan pengembangan aplikasi kita.

Sehingga fitur *seeder* bisa kita pakai dalam membuat sebuah aplikasi web.

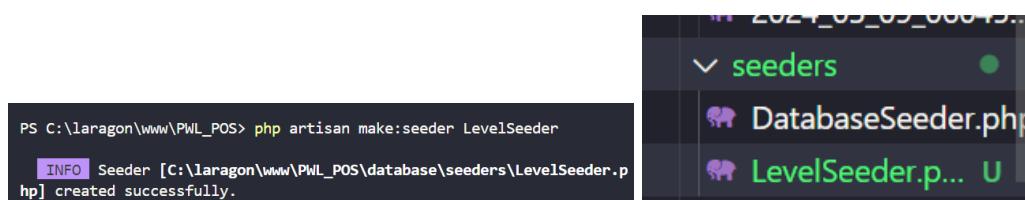
Praktikum 3 – Membuat file seeder

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

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



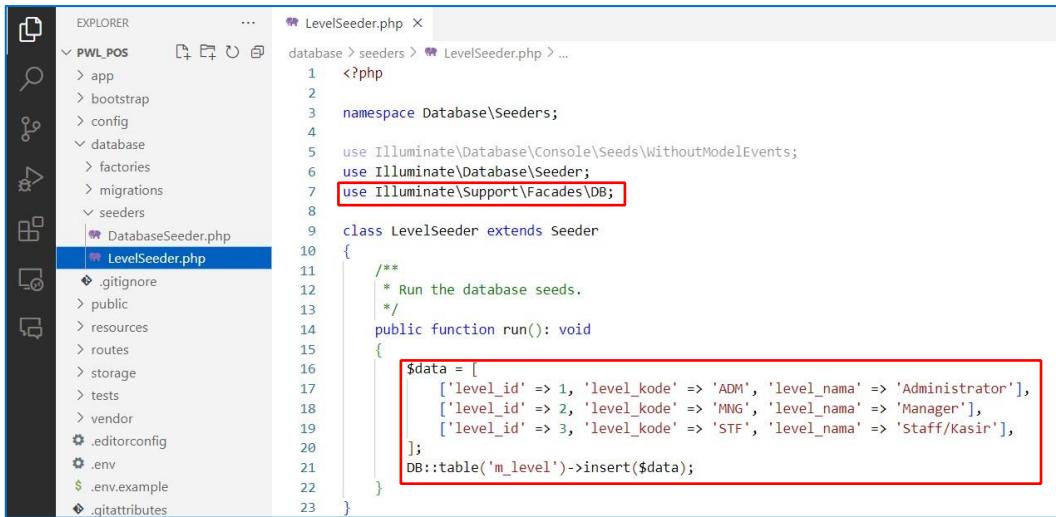
```
LevelSeeder.php
database > seeders > LevelSeeder.php > ...
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7
8  class LevelSeeder extends Seeder
9  {
10     /**
11      * Run the database seeds.
12      */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



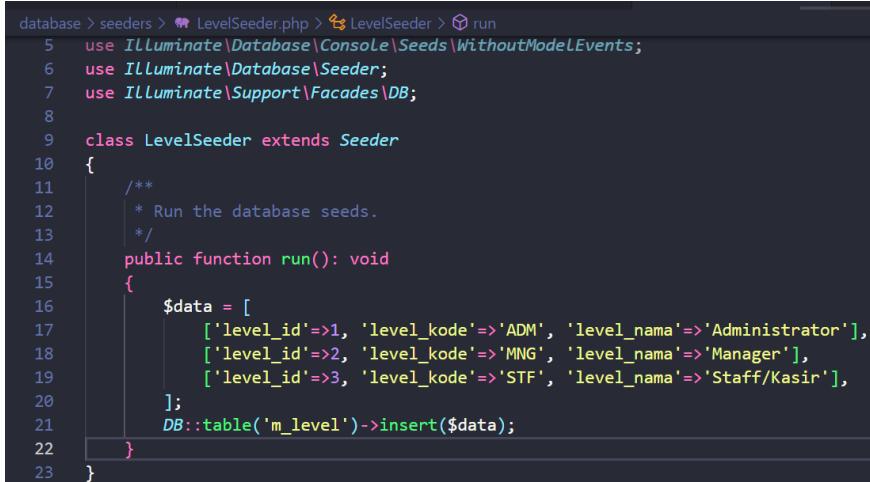
```
PS C:\laragon\www\PWL_POS> php artisan make:seeder LevelSeeder
[INFO] Seeder [C:\laragon\www\PWL_POS\database\seeders\LevelSeeder.php] created successfully.
```



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



```
database > seeders > LevelSeeder.php > LevelSeeder > run
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```



```
database > seeders > LevelSeeder.php > LevelSeeder > run
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```

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

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



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\_laragon\www\PWL_POS> php artisan db:seed --class=LevelSeeder
[INFO] Seeding database.
PS D:\_laragon\www\PWL_POS>
```



```
PS C:\laragon\www\PWL_POS> php artisan db:seed --class=LevelSeeder
INFO Seeding database.
```

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

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

With selected: Check all Edit Copy Delete Export

SELECT * FROM `m_level`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

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

Extra options

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

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

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

```
PS C:\laragon\www\PWL_POS> php artisan make:seeder UserSeeder
INFO Seeder [C:\laragon\www\PWL_POS\database\seeders\UserSeeder.php] created successfully.
```

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 }
```

```
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Support\Facades\DB;
9
10 class UserSeeder extends Seeder
11 {
12     /**
13      * Run the database seeds.
14      */
15     public function run(): void
16     {
17         $data = [
18             [
19                 'user_id' => 1,
20                 'level_id' => 1,
21                 'username' => 'admin',
22                 'nama' => 'Administrator',
23                 'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
24             ],
25             [
26                 'user_id' => 2,
27                 'level_id' => 2,
28                 'username' => 'manager',
29                 'nama' => 'Manager',
30                 'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
31             ],
32             [
33                 'user_id' => 3,
34                 'level_id' => 3,
35                 'username' => 'staff',
36                 'nama' => 'Staff/Kasir',
37                 'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
38             ],
39         ];
40         DB::table('m_user')->insert($data);
41     }
42 }
```



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"/>	Edit Copy Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6IXV...
<input type="checkbox"/>	Edit Copy Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhIFaruLxkh5wvZ9NGRp...
<input type="checkbox"/>	Edit Copy Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMy/BHnbJ9W...

SELECT * FROM `m_user`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

[Extra options](#)

		user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	admin	Administrator	\$2y\$12\$uRLgTycCq1YK4.kLTs3TeeihTsOPjEeOp0mUtuDWiu...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	2	manager	Manager	\$2y\$12\$4MSbI4wLMioNALc8dzZBO9wISXVge0gNyGnaQNhKw7...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	3	staff	Staff/Kasir	\$2y\$12\$FFTILtLca9k3XoTlwCda/u5XLfo9nRBGLdPsWMny2H...	NULL	NULL

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

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

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

[m_kategori](#)

```
PS C:\laragon\www\PWL_POS> php artisan make:seeder kategoriSeeder
[INFO] Seeder [C:\laragon\www\PWL_POS\database\seeders\kategoriSeeder.php] created successfully.
```



```
database > seeders > kategoriSeeder.php > kategoriSeeder > run
  9  class kategoriSeeder extends Seeder
 10  {
 11      /**
 12       * Run the database seeds.
 13      */
 14      public function run(): void
 15      {
 16          $data = [
 17              [
 18                  'kategori_id' => 1,
 19                  'level_kode' => 'k1',
 20                  'kategori_nama' => 'Makanan',
 21              ],
 22              [
 23                  'kategori_id' => 2,
 24                  'level_kode' => 'k2',
 25                  'kategori_nama' => 'Minuman',
 26              ],
 27              [
 28                  'kategori_id' => 3,
 29                  'level_kode' => 'k3',
 30                  'kategori_nama' => 'Sembako',
 31              ],
 32              [
 33                  'kategori_id' => 4,
 34                  'level_kode' => 'k4',
 35                  'kategori_nama' => 'Body Care',
 36              ],
 37              [
 38                  'kategori_id' => 5,
 39                  'level_kode' => 'k5',
 40                  'kategori_nama' => 'Alat Kebersihan',
 41              ],
 42          ];
 43          DB::table('m_kategori')->insert($data);
 44      }
 45  }
```

PS C:\laragon\www\PWL_POS> php artisan db:seed --class=kategoriSeeder

INFO Seeding database.

SELECT * FROM 'm_kategori'					
<input type="checkbox"/> Profiling <input type="checkbox"/> Edit inline <input type="checkbox"/> Explain SQL <input type="checkbox"/> Create PHP code <input type="checkbox"/> Refresh					
<input type="checkbox"/> Show all <input type="checkbox"/> Number of rows: 25 <input type="checkbox"/> Filter rows <input type="checkbox"/> Search this table <input type="checkbox"/> Sort by key: None					
Extra options					
<input type="checkbox"/> kategori_id level_kode kategori_nama created_at updated_at					
<input type="checkbox"/> 1 k1 Makanan NULL NULL					
<input type="checkbox"/> 2 k2 Minuman NULL NULL					
<input type="checkbox"/> 3 k3 Sembako NULL NULL					
<input type="checkbox"/> 4 k4 Body Care NULL NULL					
<input type="checkbox"/> 5 k5 Alat Kebersihan NULL NULL					

m_barang

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

INFO Seeder [C:\laragon\www\PWL_POS\database\seeders\BarangSeeder.php] created successfully.



```
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class BarangSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13     */
14     public function run(): void
15     {
16         $data = [
17             [
18                 'barang_id' => 1,
19                 'kategori_id' => 1,
20                 'barang_kode' => 'b1',
21                 'barang_nama' => 'Sosis',
22                 'harga_beli' => 1000,
23                 'harga_jual' => 2000,
24             ],
25             [
26                 'barang_id' => 2,
27                 'kategori_id' => 1,
28                 'barang_kode' => 'b2',
29                 'barang_nama' => 'Mi Goreng',
30                 'harga_beli' => 3000,
31                 'harga_jual' => 4000,
32             ],
33             [
34                 'barang_id' => 3,
35                 'kategori_id' => 2,
36                 'barang_kode' => 'b3',
37                 'barang_nama' => 'Nutrisari',
38                 'harga_beli' => 5000,
39                 'harga_jual' => 6000,
40             ],
41             [
42                 'barang_id' => 4,
43                 'kategori_id' => 2,
44                 'barang_kode' => 'b4',
45                 'barang_nama' => 'Aqua',
46                 'harga_beli' => 2500,
47                 'harga_jual' => 3000,
48             ],
49             [
50                 'barang_id' => 5,
51                 'kategori_id' => 3,
52                 'barang_kode' => 'b5',
53                 'barang_nama' => 'Beras',
54                 'harga_beli' => 15000,
55                 'harga_jual' => 17000,
56             ],
57         ];
58         [
59             [
60                 'barang_id' => 6,
61                 'kategori_id' => 3,
62                 'barang_kode' => 'b6',
63                 'barang_nama' => 'Gula',
64                 'harga_beli' => 10000,
65                 'harga_jual' => 12000,
66             ],
67             [
68                 'barang_id' => 7,
69                 'kategori_id' => 4,
70                 'barang_kode' => 'b7',
71                 'barang_nama' => 'Sabun',
72                 'harga_beli' => 3000,
73                 'harga_jual' => 4000,
74             ],
75             [
76                 'barang_id' => 8,
77                 'kategori_id' => 4,
78                 'barang_kode' => 'b8',
79                 'barang_nama' => 'Pasta Gigi',
80                 'harga_beli' => 5000,
81                 'harga_jual' => 6000,
82             ],
83             [
84                 'barang_id' => 9,
85                 'kategori_id' => 5,
86                 'barang_kode' => 'b9',
87                 'barang_nama' => 'Sapu',
88                 'harga_beli' => 10000,
89                 'harga_jual' => 12000,
90             ],
91             [
92                 'barang_id' => 10,
93                 'kategori_id' => 5,
94                 'barang_kode' => 'b10',
95                 'barang_nama' => 'Cikrak',
96                 'harga_beli' => 7000,
97                 'harga_jual' => 8000,
98             ],
99         ];
100     }
101 }

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

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

INFO Seeding database.

SELECT * FROM `m_barang`								
<input type="checkbox"/> Profiling <input type="checkbox"/> Edit inline <input type="checkbox"/> Edit <input type="checkbox"/> Explain SQL <input type="checkbox"/> Create PHP code <input type="checkbox"/> Refresh								
<input type="checkbox"/> Show all <input type="checkbox"/> Number of rows: 25 <input type="checkbox"/> Filter rows: Search this table Sort by key: None								
<input type="checkbox"/> Extra options								
← T →	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	1	b1	Sosis	1000	2000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	1	b2	Mi Goreng	3000	4000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	2	b3	Nutrisari	5000	6000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	2	b4	Aqua	2500	3000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	5	3	b5	Beras	15000	17000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	6	3	b6	Gula	10000	12000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	7	4	b7	Sabun	3000	4000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	8	4	b8	Pasta Gigi	5000	6000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	9	5	b9	Sapu	10000	12000	NULL	NULL
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	10	5	b10	Cikrak	7000	8000	NULL	NULL



t_stok

```
PS C:\laragon\www\PWL_POS> php artisan make:seeder StokSeeder
[INFO] Seeder [C:\laragon\www\PWL_POS\database\seeders\StokSeeder.php] created successfully.

6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class StokSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             [
18                 'stok_id' => 1,
19                 'barang_id' => 1,
20                 'user_id' => 3,
21                 'stok_tanggal' => "2024-03-09 14:11:00.",
22                 'stok_jumlah' => '10',
23             ],
24             [
25                 'stok_id' => 2,
26                 'barang_id' => 2,
27                 'user_id' => 3,
28                 'stok_tanggal' => "2024-03-09 14:11:00.",
29                 'stok_jumlah' => '10',
30             ],
31             [
32                 'stok_id' => 3,
33                 'barang_id' => 3,
34                 'user_id' => 3,
35                 'stok_tanggal' => "2024-03-09 14:11:00.",
36                 'stok_jumlah' => '10',
37             ],
38             [
39                 'stok_id' => 4,
40                 'barang_id' => 4,
41                 'user_id' => 3,
42                 'stok_tanggal' => "2024-03-09 14:11:00.",
43                 'stok_jumlah' => '10',
44             ],
45             [
46                 'stok_id' => 5,
47                 'barang_id' => 5,
48                 'user_id' => 3,
49                 'stok_tanggal' => "2024-03-09 14:11:00.",
50                 'stok_jumlah' => '10',
51             ],
52             [
53                 'stok_id' => 6,
54                 'barang_id' => 6,
55                 'user_id' => 3,
56                 'stok_tanggal' => "2024-03-09 14:11:00.",
57                 'stok_jumlah' => '10',
58             ],
59             [
60                 'stok_id' => 7,
61                 'barang_id' => 7,
62                 'user_id' => 3,
63                 'stok_tanggal' => "2024-03-09 14:11:00.",
64                 'stok_jumlah' => '10',
65             ],
66         ];
67         [
68             [
69                 'stok_id' => 8,
70                 'barang_id' => 8,
71                 'user_id' => 3,
72                 'stok_tanggal' => "2024-03-09 14:11:00.",
73                 'stok_jumlah' => '10',
74             ],
75             [
76                 'stok_id' => 9,
77                 'barang_id' => 9,
78                 'user_id' => 3,
79                 'stok_tanggal' => "2024-03-09 14:11:00.",
80                 'stok_jumlah' => '10',
81             ],
82             [
83                 'stok_id' => 10,
84                 'barang_id' => 10,
85                 'user_id' => 3,
86                 'stok_tanggal' => "2024-03-09 14:11:00.",
87                 'stok_jumlah' => '10',
88             ],
89         ];
90     }
91 }

PS C:\laragon\www\PWL_POS> php artisan db:seed --class=StokSeeder
[INFO] Seeding database.
```



SELECT * FROM `t_stok`

Profiling | Edit inline | [Edit] | Explain SQL | Create PHP code | [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	1	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	2	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	3	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	4	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	5	5	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	6	6	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	7	7	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	8	8	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	9	9	3	2024-03-09 14:11:00	10	NULL	NULL
<input type="checkbox"/> Edit <input type="button" value="Copy"/> <input type="button" value="Delete"/>	10	10	3	2024-03-09 14:11:00	10	NULL	NULL

t_penjualan

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

```
[INFO] Seeder [C:\laragon\www\PWL_POS\database\seeders\PenjualanSeeder.php] created successfully.
```

```
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class PenjualanSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             [
18                 'penjualan_id' => 1,
19                 'user_id' => 3,
20                 'pembeli' => 'Dea',
21                 'penjualan_kode' => 'p1',
22                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
23             ],
24             [
25                 'penjualan_id' => 2,
26                 'user_id' => 3,
27                 'pembeli' => 'Ratna',
28                 'penjualan_kode' => 'p2',
29                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
30             ],
31             [
32                 'penjualan_id' => 3,
33                 'user_id' => 3,
34                 'pembeli' => 'Elva',
35                 'penjualan_kode' => 'p3',
36                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
37             ],
38             [
39                 'penjualan_id' => 4,
40                 'user_id' => 3,
41                 'pembeli' => 'Ana',
42                 'penjualan_kode' => 'p4',
43                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
44             ],
45             [
46                 'penjualan_id' => 5,
47                 'user_id' => 3,
48                 'pembeli' => 'Jihan',
49                 'penjualan_kode' => 'p5',
50                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
51             ],
52             [
53                 'penjualan_id' => 6,
54                 'user_id' => 3,
55                 'pembeli' => 'Fanes',
56                 'penjualan_kode' => 'p6',
57                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
58             ],
59             [
60                 'penjualan_id' => 7,
61                 'user_id' => 3,
62                 'pembeli' => 'Putri',
63                 'penjualan_kode' => 'p7',
64                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
65             ],

```

```
66             [
67                 'penjualan_id' => 8,
68                 'user_id' => 3,
69                 'pembeli' => 'Nadila',
70                 'penjualan_kode' => 'p8',
71                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
72             ],
73             [
74                 'penjualan_id' => 9,
75                 'user_id' => 3,
76                 'pembeli' => 'Octa',
77                 'penjualan_kode' => 'p9',
78                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
79             ],
80             [
81                 'penjualan_id' => 10,
82                 'user_id' => 3,
83                 'pembeli' => 'Maulita',
84                 'penjualan_kode' => 'p10',
85                 'penjualan_tanggal' => "2024-03-09 14:19:00.",
86             ],
87         ];
88         DB::table('t_penjualan')->insert($data);
89     }
90 }
```

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

```
[INFO] Seeding database.
```



SELECT * FROM `t_penjualan`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/>	1	3	Dea	p1	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	2	3	Ratna	p2	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	3	3	Eiva	p3	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	4	3	Ana	p4	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	5	3	Jihan	p5	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	6	3	Fanes	p6	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	7	3	Putri	p7	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	8	3	Nadila	p8	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	9	3	Octa	p9	2024-03-09 14:19:00	NULL	NULL
<input type="checkbox"/>	10	3	Maulita	p10	2024-03-09 14:19:00	NULL	NULL

t_penjualan_detail

```
PS C:\laragon\www\PWL_POS> php artisan make:seeder PenjualanDetailSeeder
[INFO] Seeder [C:\laragon\www\PWL_POS\database\seeders\PenjualanDetailSeeder.php] created successfully.
```

```
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class PenjualanDetailSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             [
18                 'detail_id' => 1,
19                 'penjualan_id' => 1,
20                 'barang_id' => 2,
21                 'harga' => 4000,
22                 'jumlah' => 2,
23             ],
24             [
25                 'detail_id' => 2,
26                 'penjualan_id' => 1,
27                 'barang_id' => 10,
28                 'harga' => 8000,
29                 'jumlah' => 1,
30             ],
31             [
32                 'detail_id' => 3,
33                 'penjualan_id' => 1,
34                 'barang_id' => 5,
35                 'harga' => 17000,
36                 'jumlah' => 1,
37             ],
38             [
39                 'detail_id' => 4,
40                 'penjualan_id' => 2,
41                 'barang_id' => 8,
42                 'harga' => 6000,
43                 'jumlah' => 1,
44             ],
45             [
46                 'detail_id' => 5,
47                 'penjualan_id' => 2,
48                 'barang_id' => 1,
49                 'harga' => 2000,
50                 'jumlah' => 4,
51             ],
52             [
53                 'detail_id' => 6,
54                 'penjualan_id' => 2,
55                 'barang_id' => 2,
56                 'harga' => 4000,
57                 'jumlah' => 5,
58             ],
59             [
60                 'detail_id' => 7,
61                 'penjualan_id' => 3,
62                 'barang_id' => 4,
63                 'harga' => 3000,
64                 'jumlah' => 7,
65             ],

```



```
66
67     [ 94 ▼
68         'detail_id' => 8, 95
69         'penjualan_id' => 3, 96
70         'barang_id' => 7, 97
71         'harga' => 4000, 98
72         'jumlah' => 1, 99
73     ], 100
74     [
75         'detail_id' => 9, 101 ▼
76         'penjualan_id' => 3, 102
77         'barang_id' => 9, 103
78         'harga' => 12000, 104
79         'jumlah' => 1, 105
80     ], 106
81     [
82         'detail_id' => 10, 107
83         'penjualan_id' => 4, 108 ▼
84         'barang_id' => 6, 109
85         'harga' => 12000, 110
86         'jumlah' => 1, 111
87     ], 112
88     [
89         'detail_id' => 11, 113
90         'penjualan_id' => 4, 114
91         'barang_id' => 3, 115 ▼
92         'harga' => 6000, 116
93         'jumlah' => 5, 117
94     ], 118
95
96     [
97         'detail_id' => 12, 119
98         'penjualan_id' => 4, 120
99         'barang_id' => 1, 121
100
101     ], 122 ▼
102     [
103         'detail_id' => 13, 123
104         'penjualan_id' => 5, 124
105         'barang_id' => 5, 125
106         'harga' => 17000, 126
107         'jumlah' => 2, 127
108     ], 128
109     [
110         'detail_id' => 14, 129 ▼
111         'penjualan_id' => 5, 130
112         'barang_id' => 10, 131
113         'harga' => 8000, 132
114         'jumlah' => 1, 133
115     ], 134
116     [
117         'detail_id' => 15, 135
118         'penjualan_id' => 5, 136 ▼
119         'barang_id' => 2, 137
120         'harga' => 4000, 138
121         'jumlah' => 4, 139
122     ], 140
123     [
124         'detail_id' => 16, 141
125         'penjualan_id' => 2, 142
126         'barang_id' => 2, 143 ▼
127         'harga' => 4000, 144
128         'jumlah' => 2, 145
129     ], 146
130     [
131         'detail_id' => 17, 147
132         'penjualan_id' => 6, 148
133         'barang_id' => 10, 149
134         'harga' => 8000, 150
135         'jumlah' => 1, 151
136     ], 152
137     [
138         'detail_id' => 18, 153
139         'penjualan_id' => 6, 154
140         'barang_id' => 5, 155
141         'harga' => 17000, 156
142         'jumlah' => 1, 157
143     ], 158
144     [
145         'detail_id' => 19, 159
146         'penjualan_id' => 7, 160
147         'barang_id' => 8, 161
148         'harga' => 6000, 162
149         'jumlah' => 1, 163
150     ], 164
```

```
150 ▼ [ 178 ▼ [
151   'detail_id' => 20,
152   'penjualan_id' => 7,
153   'barang_id' => 1,
154   'harga' => 2000,
155   'jumlah' => 4,
156 ],
157 [
158   'detail_id' => 21,
159   'penjualan_id' => 7,
160   'barang_id' => 2,
161   'harga' => 4000,
162   'jumlah' => 5,
163 ],
164 [
165   'detail_id' => 22,
166   'penjualan_id' => 8,
167   'barang_id' => 4,
168   'harga' => 3000,
169   'jumlah' => 7,
170 ],
171 [
172   'detail_id' => 23,
173   'penjualan_id' => 8,
174   'barang_id' => 7,
175   'harga' => 4000,
176   'jumlah' => 1,
177 ],
178 [
179   'detail_id' => 24,
180   'penjualan_id' => 8,
181   'barang_id' => 9,
182   'harga' => 12000,
183   'jumlah' => 1,
184 ],
185 [
186   'detail_id' => 25,
187   'penjualan_id' => 9,
188   'barang_id' => 6,
189   'harga' => 12000,
190   'jumlah' => 1,
191 ],
192 [
193   'detail_id' => 26,
194   'penjualan_id' => 9,
195   'barang_id' => 3,
196   'harga' => 6000,
197   'jumlah' => 5,
198 ],
199 [
200   'detail_id' => 27,
201   'penjualan_id' => 9,
202   'barang_id' => 1,
203   'harga' => 2000,
204   'jumlah' => 12,
205 ],
206 [
207   'detail_id' => 28,
208   'penjualan_id' => 10,
209   'barang_id' => 5,
210   'harga' => 17000,
211   'jumlah' => 2,
212 ],
213 [
214   'detail_id' => 29,
215   'penjualan_id' => 10,
216   'barang_id' => 10,
217   'harga' => 8000,
218   'jumlah' => 1,
219 ],
220 [
221   'detail_id' => 30,
222   'penjualan_id' => 10,
223   'barang_id' => 2,
224   'harga' => 4000,
225   'jumlah' => 4,
226 ],
227 ];
DB::table('t_penjualan_detail')->insert($data);
```



The screenshot shows a MySQL query results page. The query is:

```
SELECT * FROM `t_penjualan_detail`
```

The results table has columns: detail_id, penjualan_id, barang_id, harga, jumlah, created_at, updated_at. The data is as follows:

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>	1	1	2	4000	2	NULL	NULL
<input type="checkbox"/>	2	1	10	8000	1	NULL	NULL
<input type="checkbox"/>	3	1	5	17000	1	NULL	NULL
<input type="checkbox"/>	4	2	8	6000	1	NULL	NULL
<input type="checkbox"/>	5	2	1	2000	4	NULL	NULL
<input type="checkbox"/>	6	2	2	4000	5	NULL	NULL
<input type="checkbox"/>	7	3	4	3000	7	NULL	NULL
<input type="checkbox"/>	8	3	7	4000	1	NULL	NULL
<input type="checkbox"/>	9	3	9	12000	1	NULL	NULL
<input type="checkbox"/>	10	4	6	12000	1	NULL	NULL
<input type="checkbox"/>	11	4	3	6000	5	NULL	NULL
<input type="checkbox"/>	12	4	1	2000	12	NULL	NULL
<input type="checkbox"/>	13	5	5	17000	2	NULL	NULL

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

D. DB FACADE

DB Façade merupakan fitur dari Laravel yang digunakan untuk melakukan *query* secara langsung dengan mengetikkan 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`

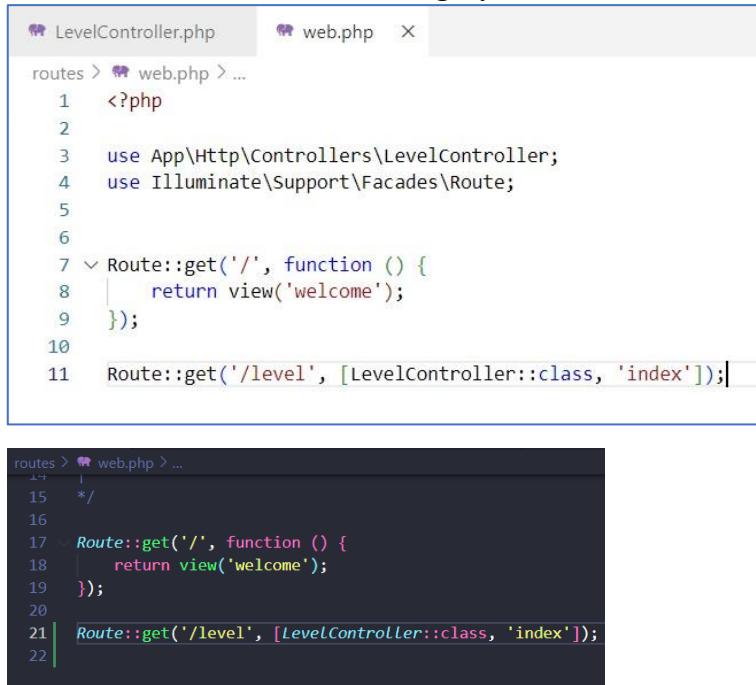
```
php artisan make:controller LevelController
```

```
PS C:\laragon\www\PWL_POS> php artisan make:controller LevelController
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\LevelController.php] created successfully.

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



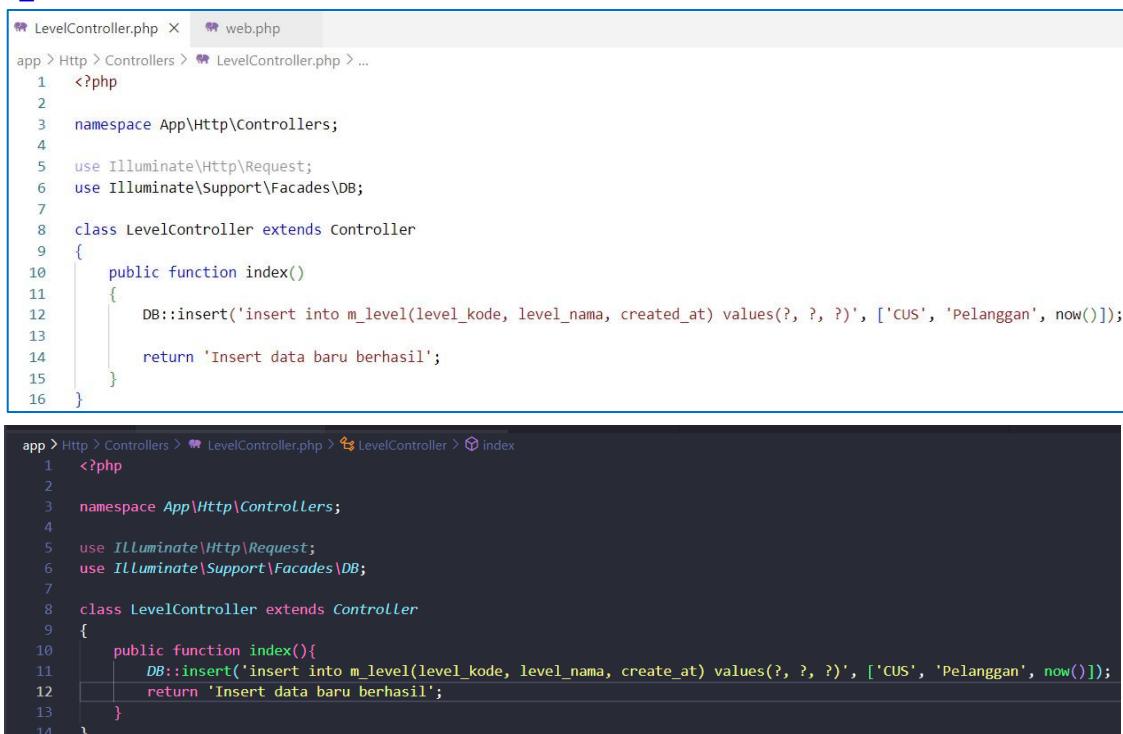
2. Kita modifikasi dulu untuk *routing*-nya, ada di `PWL_POS/routes/web.php`



```
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']);|
```

```
routes > web.php > ...
15  */
16
17 Route::get('/', function () {
18     return view('welcome');
19 });
20
21 Route::get('/level', [LevelController::class, 'index']);|
```

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



```
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 }
```

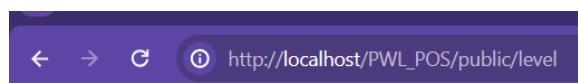
```
app > Http > Controllers > LevelController.php > LevelController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12         DB::insert('insert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

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



table `m_level`

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



Insert data baru berhasil

SELECT * FROM `m_level`						
<input type="checkbox"/> Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]						
<input type="checkbox"/> Show all Number of rows: 25 ▾						
Extra options						
	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	<input type="button" value="Edit"/> <input type="button" value="Copy"/> <input type="button" value="Delete"/>	4	CUS	Pelanggan	2024-03-09 07:56:18	NULL

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

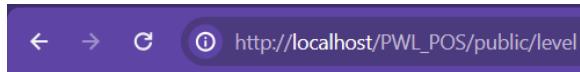
```
LevelController.php x web.php
app > Http > Controllers > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12  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

```
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`



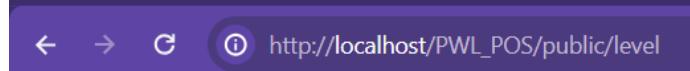
Update data berhasil. Jumlah data yang diupdate: 1 baris

SELECT * FROM `m_level`					
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]					
<input type="checkbox"/> Show all Number of rows: 25 <input type="button" value="▼"/>					
Extra options					
	Edit	Copy	Delete	level_id	level_kode
	Edit	Copy	Delete	1	ADM
	Edit	Copy	Delete	2	MNG
	Edit	Copy	Delete	3	STF
	Edit	Copy	Delete	4	CUS

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 }
```

```
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     }
}
```



Delete data berhasil. Jumlah data yang dihapus: 1 baris



The screenshot shows a MySQL query results page. The query is:

```
SELECT * FROM `m_level`
```

The results table has columns: level_id, level_kode, level_nama, created_at, updated_at. There are three rows:

	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

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

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

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

PWL_POS/resources/view/level.blade.php

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



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

10. Silahkan dicoba pada browser dan amati apa yang terjadi



Data Level Pengguna

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

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

<i>Method Query Builder</i>	<i>Query yang dihasilkan</i>
<code>DB::table('m_kategori')->get();</code>	<code>select * from m_kategori</code>



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

```
PS C:\laragon\www\PWL_POS> php artisan make:controller KategoriController
[INFO] Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\KategoriController.php] created successfully.
```

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']);
```

```
routes > web.php > ...
10
11 */
12
13 Route::get('/', function () {
14     return view('welcome');
15 });
16
17 Route::get('/level', [LevelController::class, 'index']);
18 Route::get('/kategori', [KategoriController::class, 'index']);
```

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



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

```
app > Http > Controllers > KategoriController.php > index
7
8
9 class KategoriController extends Controller
10 {
11     public function index(){
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`

← → ⌂ ⓘ http://localhost/PWL_POS/public/kategori

Insert data baru berhasil

SELECT * FROM `m_kategori`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25

Extra options

	kategori_id	level_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	1	K1	Makanan	NULL	NULL
<input type="checkbox"/>	2	K2	Minuman	NULL	NULL
<input type="checkbox"/>	3	K3	Sembako	NULL	NULL
<input type="checkbox"/>	4	K4	Body Care	NULL	NULL
<input type="checkbox"/>	5	K5	Alat Kebersihan	NULL	NULL
<input type="checkbox"/>	6	SNK	Snack/Makanan Ringan	2024-03-09 08:31:04	NULL



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

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

```
app > Http > Controllers >  KategoriController.php >  KategoriController >  index
9  class KategoriController extends Controller
10 {
11     public function index(){
12         // $data = [
13             // 'level.kode' => 'SNK',
14             // 'kategori.nama' => 'Snack/Makanan Ringan',
15             // 'created_at' => now()
16         // ];
17         // DB::table('m_kategori')->insert($data);
18         // return 'Insert data baru berhasil';
19
20         $row = DB::table('m+kategori')->where('level.kode', 'SNK')->update(['kategori.nama' => 'Camilan']);
21         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22     }
23 }
```

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



Update data berhasil. Jumlah data yang diupdate: 1 baris



SELECT * FROM `m_kategori`						
<input type="checkbox"/> Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]						
<input type="checkbox"/> Show all Number of rows: 25						
Extra options						
← T →	▼	kategori_id	level_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>		1	k1	Makanan	NULL	NULL
<input type="checkbox"/>		2	k2	Minuman	NULL	NULL
<input type="checkbox"/>		3	k3	Sembako	NULL	NULL
<input type="checkbox"/>		4	k4	Body Care	NULL	NULL
<input type="checkbox"/>		5	k5	Alat Kebersihan	NULL	NULL
<input type="checkbox"/>		6	SNK	Camilan	2024-03-09 08:31:04	NULL

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

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

```
app > Http > Controllers > 🐾 KategoriController.php > 📁 KategoriController > ⚒ index
  9  class KategoriController extends Controller
10  {
11      public function index(){
12          // $data = [
13          //     'level.kode' => 'SNK',
14          //     'kategori.nama' => 'Snack/Makanan Ringan',
15          //     'created_at' => now()
16          // ];
17          // DB::table('m_kategori')->insert($data);
18          // return 'Insert data baru berhasil';
19
20          // $row = DB::table('m_kategori')->where('level.kode', 'SNK')->update(['kategori.nama' => 'Camilan']);
21          // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row .' baris';
22
23          $row = DB::table('m_kategori')->where('level.kode', 'SNK')->delete();
24          return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row .' baris';
25      }
26  }
```

← → ⌂ ⏷ http://localhost/PWL_POS/public/kategori

Delete data berhasil. Jumlah data yang dihapus: 1 baris



SELECT * FROM `m_kategori`					
<input type="checkbox"/> Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code]					
<input type="checkbox"/> Show all Number of rows: 25 ▾					
Extra options					
← T →	▼ kategori_id	level_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	1	k1	Makanan	NULL	NULL
<input type="checkbox"/>	2	k2	Minuman	NULL	NULL
<input type="checkbox"/>	3	k3	Sembako	NULL	NULL
<input type="checkbox"/>	4	k4	Body Care	NULL	NULL
<input type="checkbox"/>	5	k5	Alat Kebersihan	NULL	NULL

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

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

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

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



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

```
resources > views > 🗂️ kategori.blade.php > ⚒️ html
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 Katgori</th>
13             </tr>
14             @foreach ($data as $d)
15             <tr>
16                 <td>{{ $d->kategori_id }}</td>
17                 <td>{{ $d->level_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.

← → ⌛ ⓘ http://localhost/PWL_POS/public/kategori

Data Kategori Barang

ID	Kode Kategori	Nama Katgori
1	k1	Makanan
2	k2	Minuman
3	k3	Sembako
4	k4	Body Care
5	k5	Alat Kebersihan

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>
```

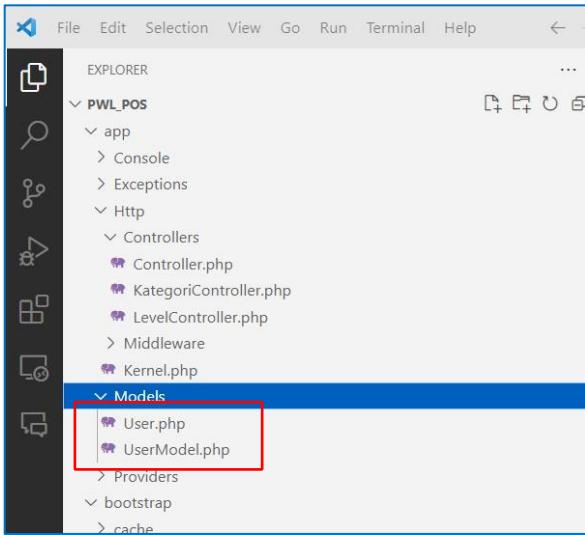
Untuk bisa melakukan operasi ***CRUD*** (*create, read/retrieve, update, delete*), kita harus membuat sebuah model sesuai dengan target tabel yang ingin digunakan. Jadi, **dalam 1 model, merepresentasikan 1 tabel database.**

Praktikum 6 – Implementasi Eloquent ORM

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



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



```
PS C:\laragon\www\PWL_POS> php artisan make:model UserModel
[INFO] Model [C:\laragon\www\PWL_POS\app\Models\UserModel.php] created successfully.
```

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

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



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

4. Kita modifikasi route web.php untuk mencoba routing ke controller [UserController](#)

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use App\Http\Controllers\UserController;
6  use Illuminate\Support\Facades\Route;
7
8
9  Route::get('/', function () {
10     return view('welcome');
11 });
12
13 Route::get('/level', [LevelController::class, 'index']);
14 Route::get('/kategori', [KategoriController::class, 'index']);
15 Route::get('/user', [UserController::class, 'index']);
16
17
18
19
20
21
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 }
```



```
PS C:\laragon\www\PWL_POS> php artisan make:controller UserController

[INFO] Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\UserController.php] created successfully.

app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\UserModel;
7
8  class UserController extends Controller
9  {
10     //coba akses model UserModel
11     public function index(){
12         $user = userModel::all(); //ambil semua data dari tabel m_user
13         return view('user', ['data'=>$user]);
14     }
15
16 }
17
```

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>
```



```
resources > views > user.blade.php > html > body > table > tr > td
 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

← → ⌛ ⏹ http://localhost/PWL_POS/public/user

Data User

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

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



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

```
app > Http > Controllers > UserController.php > UserController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\UserModel;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserController extends Controller
10 {
11
12     public function index(){
13         //tambah data user dengan Eloquent Model
14         $data = [
15             'username' => 'manager_tiga',
16             'nama' => 'Manager 3',
17             'password' => Hash::make('12345'),
18             'level_id' => 2
19         ];
20         UserModel::insert($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

← → ⌛ http://localhost/PWL_POS/public/user

Data User

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



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 }
```

```
app > Http > Controllers > UserController.php > UserController > index
6  use App\Models\UserModel;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserController extends Controller
10 {
11
12     public function index(){
13         //tambah data user dengan Eloquent Model
14         // $data = [
15             // 'username' => 'manager_tiga',
16             // 'nama' => 'Manager 3',
17             // 'password' => Hash::make('12345'),
18             // 'level_id' => 2
19             // ];
20         // UserModel::insert($data); //tambahkan data ke tabel m_user
21         $data = [
22             'nama' => 'Manager ke-3',
23         ];
24         UserModel::where('username', 'manager_tiga')->update($data); //Update data user
25
26         //coba akses model UserModel
27         $user = UserModel::all(); //ambil semua data dari tabel m_user
28         return view('user', ['data'=>$user]);
29     }
30
31 }
```

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

← → ⌛ ⓘ http://localhost/PWL_POS/public/user

Data User

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

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



G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari `APP_KEY` pada *file setting .env* Laravel?

Jawab: Fungsi APP_KEY sebagai kunci enkripsi aplikasi. Pada Laravel, file .env digunakan untuk menyimpan konfigurasi seperti pengaturan database, kredensial API, dan variable lingkungan lainnya.

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

Jawab: Dengan cara mengetikkan ‘php artisan key:generate’ di terminal

3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?

Jawab: File migrasi adalah bagian dari fitur otentifikasi bawaan dari laravel dan membantu memulai dengan cepat dalam mengembangkan aplikasi web yang membutuhkan otentifikasi pengguna

- a. `Create_users_table.php`: File migrasi ini bertanggung jawab untuk membuat tabel pengguna (users) di database. Tabel ini biasanya digunakan untuk menyimpan informasi pengguna seperti nama, email, dan password hash
- b. `create_password_resets_table.php`: File migrasi ini bertanggung jawab untuk membuat tabel password_resets. Tabel ini digunakan untuk menyimpan token reset password yang dihasilkan saat pengguna meminta reset password. Dengan token ini, pengguna dapat mengonfirmasi permintaan reset password dan membuat kata sandi baru

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

Jawab: Untuk menambahkan 2 kolom pada tabel secara otomatis

- a. `Created_at` : Kolom ini digunakan untuk menyimpan tanggal dan waktu sebuah record pertama kali dibuat (inserted) ke dalam tabel
- b. `Update_at`: Kolom ini digunakan untuk menyimpan tanggal dan waktu saat sebuah rekaman terakhir kali diperbarui (update) dalam tabel



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

Jawab: Menghasilkan tipe data bigIncrement

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

Jawab:

- `$table->id():` Jika menggunakan `$table->id()`, Laravel secara default akan membuat kolom primary key dengan nama id. Jadi, hasil migrasi akan membuat kolom dengan nama id sebagai primary key.
- `$table->id('level_id'):` Jika menggunakan `$table->id('level_id')`, secara eksplisit memberikan nama kolom primary key, yaitu level_id. Dalam hal ini, hasil migrasi akan membuat kolom dengan nama level_id sebagai primary key

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

Jawab: : Fungsinya digunakan untuk mengatur kolom sebagai unik pada table jadi menjamin tidak adanya duplikat.

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

Jawab:

- `unsignedBigInteger('level_id')` karena kolom tersebut akan digunakan untuk menyimpan nilai dari kolom id pada tabel m_level. Kolom id pada tabel m_level adalah kolom utama dengan tipe data bigIncrements.
- Kolom `level_id` pada tabel `m_user` akan menyimpan nilai dari kolom id pada tabel `m_level`, sehingga tipe data yang digunakan adalah `unsignedBigInteger` untuk menyimpan nilai dari kolom id yang memiliki tipe data bigIncrements



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

Jawab: fungsi untuk mengolah data seperti `make()`, `check()`, `needsRehash()`, `toSafeString()`, dan `checkSafeString()`. Fungsi `make()` digunakan untuk mengolah data menjadi hash, fungsi `check()` digunakan untuk memeriksa apakah data tersebut sama dengan hash yang diberikan, dan fungsi `toSafeString()` digunakan untuk mengubah data menjadi string yang aman untuk diperbarui. Kode program `Hash::make('1234')` digunakan untuk mengolah data "1234" menjadi hash. Fungsi `make()` akan mengembalikan hasil dari proses ini sebagai string.

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

Jawab: digunakan untuk menandai bahwa kolom tersebut akan dibaca dari data yang dikirimkan ke dalam request

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

Jawab: Untuk mengatur tabel dan primary key dari User Model

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

Jawab: Lebih mudah menggunakan Query Builder. Karena syntax nya lebih ringkas dan penggunaannya hampir sama dengan DB Facade

*** Sekian, dan selamat belajar ***