

**LAPORAN PRAKTIKUM**  
**JOBSHEET 03**  
**MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM**

Disusun untuk memenuhi nilai tugas  
Mata Kuliah : Pemrograman Web Lanjut



Oleh :  
Aqueena Regita Hapsari  
2341760096  
SIB 2B  
03

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS**  
**POLITEKNIK NEGERI MALANG**  
**TAHUN AJARAN 2024/2025**

Mata Kuliah : Pemrograman Web Lanjut (PWL)  
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis  
Semester : 4 (empat) / 6 (enam)  
Pertemuan ke- : 1 (satu)

## JOBSHEET 03

### MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM

Sebelumnya kita sudah membahas mengenai *Routing*, *Controller*, dan *View* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Untuk itu, kita memerlukan teknik untuk merancang/membuat table basis data sebelum membuat aplikasi. Laravel memiliki fitur dalam pengelolaan basis data seperti, migration, seeder, model, dll.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS)*, sesuai dengan **Studi Kasus PWL.pdf**.  
Jadi kita bikin project Laravel 10 dengan nama **PWL\_POS**.

*Project PWL\_POS* akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

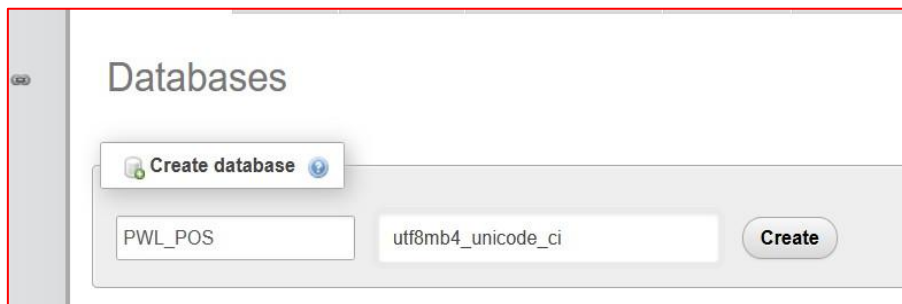
## A. PENGATURAN DATABASE

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

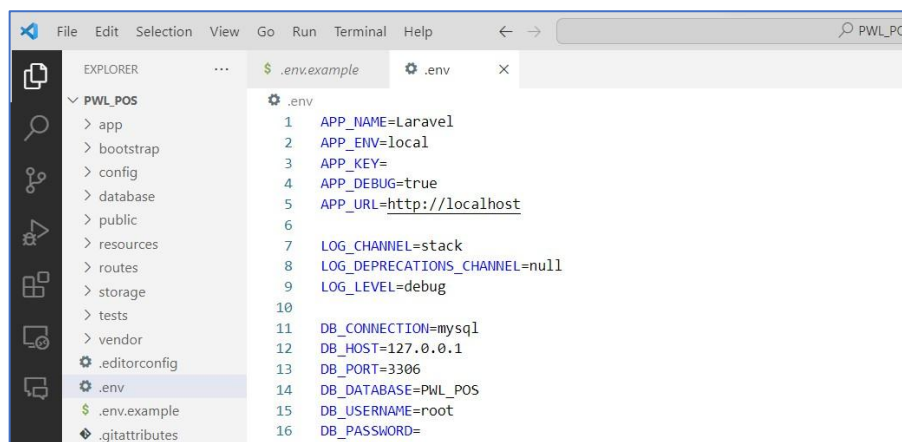
### Praktikum 1 - pengaturan database:

---

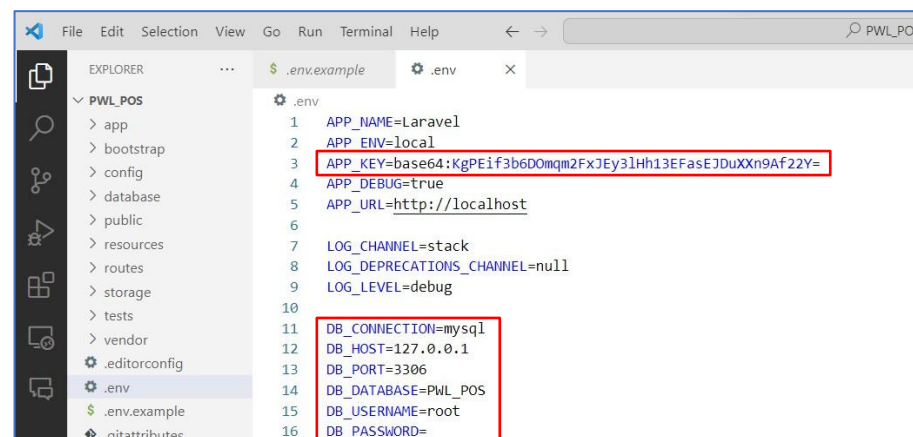
1. Buka aplikasi phpMyAdmin, dan buat database baru dengan nama **PWL\_POS**



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



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



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

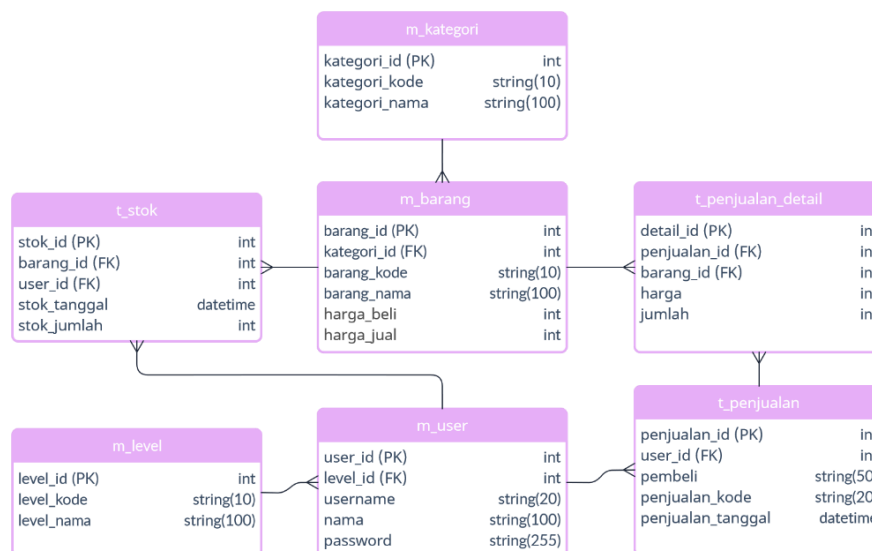
## B. MIGRATION

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

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

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

**Studi Kasus PWL.pdf**



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

### TIPS MIGRATION

Buatlah file migration untuk table yang tidak memiliki relasi (table yang tidak ada *foreign key*) dulu, dan dilanjutkan dengan membuat file migrasi yang memiliki relasi yang sedikit, dan dilanjutkan ke file migrasi dengan table yang memiliki relasi yang banyak.

Dari tips di atas, kita dapat melakukan cek untuk desain database yang sudah ada dengan mengetahui jumlah *foreign key* yang ada. Dan kita bisa menentukan table mana yang akan kita buat migrasinya terlebih dahulu.

No Urut	Nama Tabel	Jumlah FK
1	m_level	0
2	m_kategori	0
3	m_user	1
4	m_barang	1
5	t_penjualan	1
6	t_stok	2
7	t_penjualan_detail	2

### INFO

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

Pembuatan file migrasi bisa menggunakan 2 cara, yaitu

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

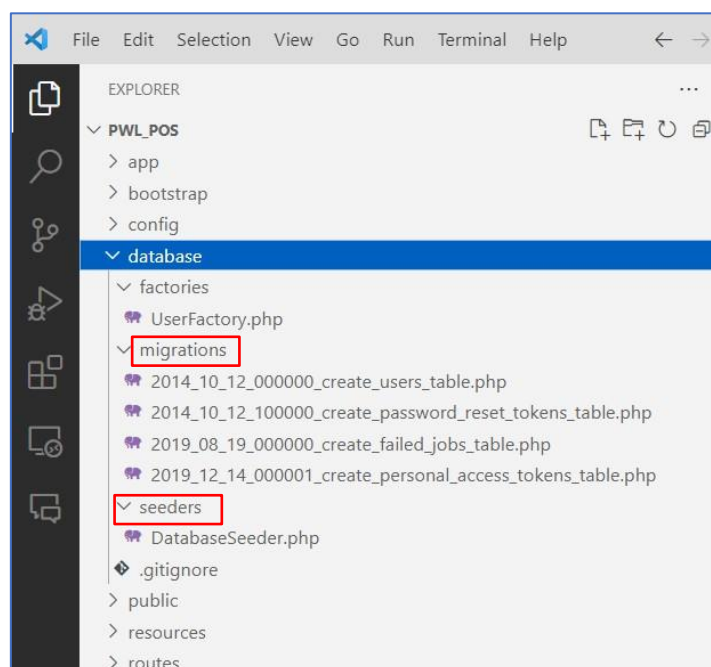
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

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

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

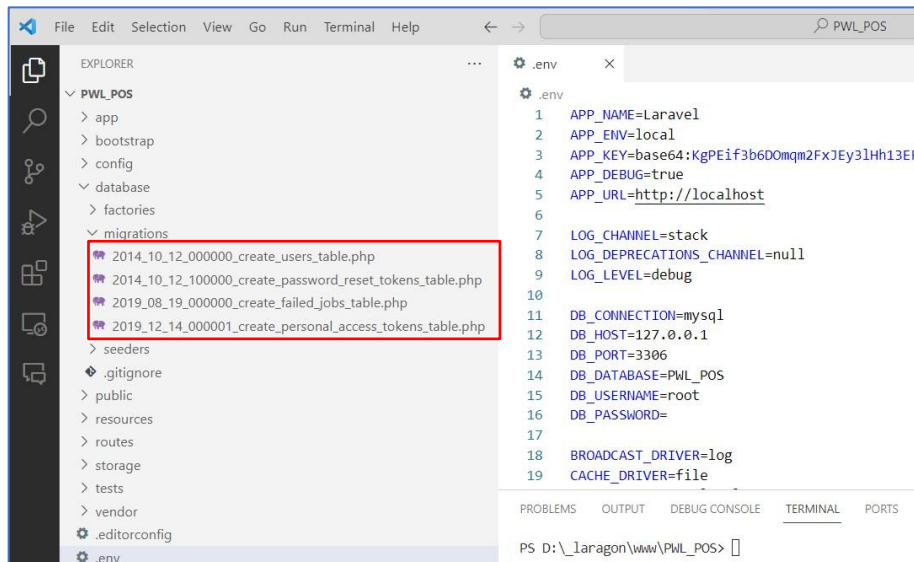
Perintah **-m** di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder **PWL\_POS/database**



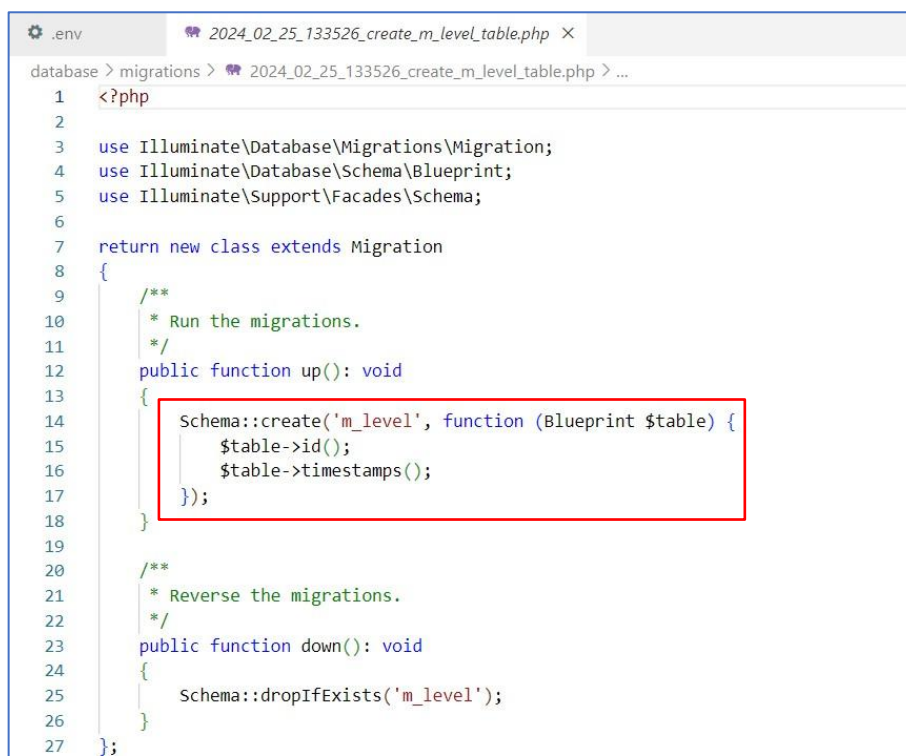
## Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



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

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



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

```

7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create('m_level', function (Blueprint $table) {
15               $table->id('level_id');
16               $table->string('level_kode', 10)->unique();
17               $table->string('level_nama', 100);
18               $table->timestamps();
19           });
20       }
21
22       /**
23       * Reverse the migrations.
24       */
25       public function down(): void
26       {
27           Schema::dropIfExists('m_level');
28       }
29 };

```

### INFO

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

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

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

```
php artisan migrate
```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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

[INFO] Preparing database.

Creating migration table ..... 12ms DONE

[INFO] Running migrations.

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

PS D:\laragon\www\PWL_POS>

```



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

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

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

**Hasil :**

```
PS D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\Mingphp artisan migrate

INFO Running migrations.

2025_03_02_182019_create_m_level_table 1,222ms DONE

PS D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\Minggu3>
```

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> items	★ Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
7 tables	Sum	7	InnoDB	utf8mb4_0900_ai_ci	112.0 KiB	0.8

```
PS D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS> php artisan migrate

INFO Running migrations.

2025_03_02_190337_create_m_kategori_table 695ms DONE
```

## Praktikum 2.2 - Pembuatan file migrasi dengan relasi

1. Buka *terminal* VSCode kalian, dan buat file migrasi untuk table **m\_user**

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



2. Buka file migrasi untuk table **m\_user**, dan modifikasi seperti berikut

```

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

```

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

**Hasil :**

```

PS D:\LARAGON\laragon\www\PemrogramanWebL
anjut-2025\PWL_POS> php artisan migrate

INFO Running migrations.

2025_03_03_022116_create_m_user_table
628ms DONE

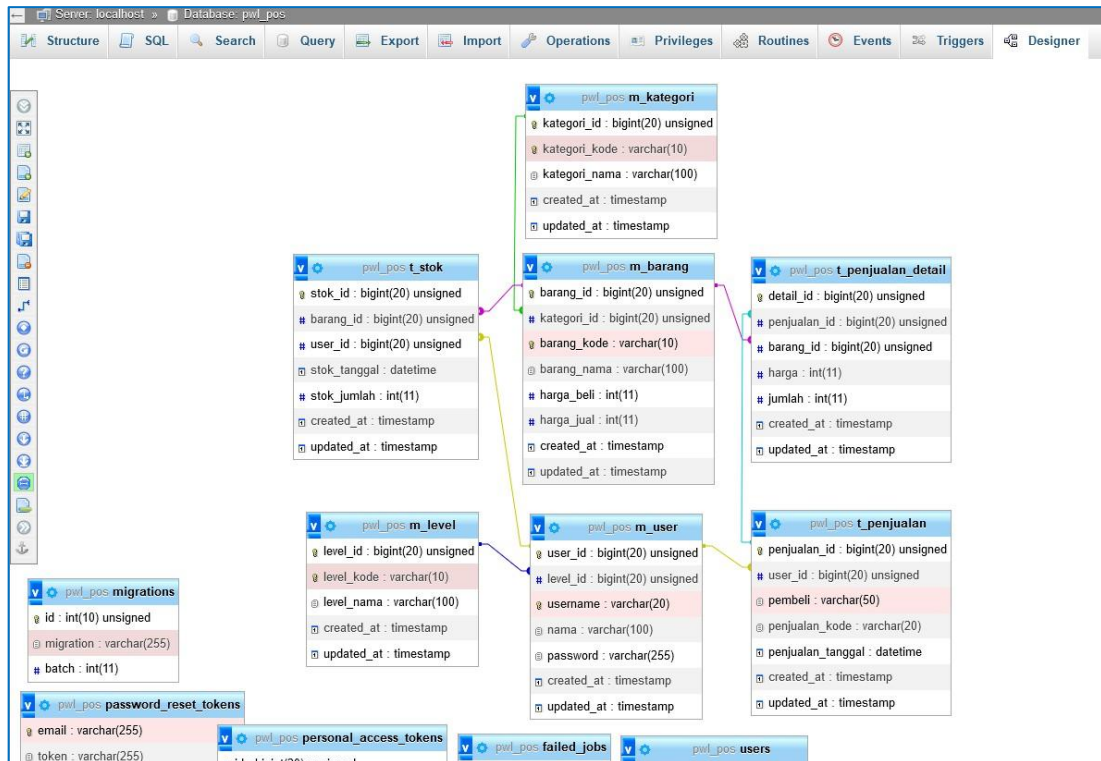
```

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs		0	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
<input type="checkbox"/> migrations		8	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
<input type="checkbox"/> m_kategori		0	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
<input type="checkbox"/> m_level		0	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
<input type="checkbox"/> m_user		0	InnoDB	utf8mb4_unicode_ci	32.0 K B	-
<input type="checkbox"/> password_reset_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
<input type="checkbox"/> personal_access_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
<input type="checkbox"/> users		0	InnoDB	utf8mb4_unicode_ci	16.0 K B	-
8 tables	Sum	8	InnoDB	utf8mb4_unicode_ci	144.0 K B	0 B

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

m_barang
t_penjualan
t_stok
t_penjualan_detail

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



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

### Hasil :

```
INFO Migration [D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS\database\migrations\2025_03_03_023124_create_m_barang_table.php] created successfully.

PS D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS> php artisan make:migration create_t_penjualan_table --table=m_penjualan

INFO Migration [D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS\database\migrations\2025_03_03_023150_create_t_penjualan_table.php] created successfully.

PS D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS> php artisan make:migration create_t_stok_table --table=m_stok

INFO Migration [D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS\database\migrations\2025_03_03_023206_create_t_stok_table.php] created successfully.

PS D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS> php artisan make:migration create_t_penjualan_detail_table --table=m_penjualan_detail

INFO Migration [D:\LARAGON\laragon\www\PemrogramanWebLanjut-2025\PWL_POS\database\migrations\2025_03_03_023215_create_t_penjualan_detail_table.p
```

## 1. Tabel m\_barang

```
Schema::create('m_barang', function (Blueprint $table) {
    $table->id('barang_id');
    $table->foreignId('kategori_id')->constrained('m_kategori')->onDelete('cascade');
    $table->string('barang_kode', 10)->unique();
    $table->string('barang_nama', 100);
    $table->integer('harga_beli');
    $table->integer('harga_jual');
    $table->timestamps();
});
```

## 2. Tabel t\_penjualan

```
Schema::create('t_penjualan', function (Blueprint $table) {
    $table->id('penjualan_id');
    $table->unsignedBigInteger('user_id'); // Foreign key yang benar
    $table->string('pembeli', 50);
    $table->string('penjualan_kode', 20);
    $table->dateTime('penjualan_tanggal');
    $table->timestamps();

    // Foreign key diperbaiki
    $table->foreign('user_id')->references('user_id')->on('m_user')->onDelete('cascade');
});
```

## 3. Tabel t\_stok

```
Schema::create('t_stok', function (Blueprint $table) {
    $table->id('stok_id');
    $table->unsignedBigInteger('barang_id');
    $table->unsignedBigInteger('user_id');
    $table->dateTime('stok_tanggal');
    $table->integer('stok_jumlah');
    $table->timestamps();

    // Foreign keys
    $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
    $table->foreign('user_id')->references('user_id')->on('m_user')->onDelete('cascade');
});
```

## 4. Tabel t\_penjualan\_detail

```
Schema::create('t_penjualan_detail', function (Blueprint $table) {
    $table->unsignedBigInteger('detail_id');
    $table->unsignedBigInteger('penjualan_id');
    $table->unsignedBigInteger('barang_id');
    $table->integer('harga');
    $table->integer('jumlah');
    $table->timestamps();

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

Table	Action	Rows	Type	Collation
<input type="checkbox"/> failed_jobs	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> migrations	Browse  Structure  Search  Insert  Empty  Drop	12	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> m_barang	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> m_kategori	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> m_level	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> m_user	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> password_reset_tokens	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> personal_access_tokens	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> t_penjualan	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> t_penjualan_detail	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> t_stok	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
<input type="checkbox"/> users	Browse  Structure  Search  Insert  Empty  Drop	0	InnoDB	utf8mb4_unicode_
12 tables	Sum	12	InnoDB	utf8mb4_unicode_

## C. SEEDER

Seeder merupakan sebuah fitur yang memungkinkan kita untuk mengisi database kita dengan data awal atau data *dummy* yang telah ditentukan. Seeder memungkinkan kita untuk membuat data awal yang sama untuk setiap penggunaan dalam pembangunan aplikasi. Umumnya, data yang sering dibuat *seeder* adalah data pengguna karena data tersebut akan digunakan saat aplikasi pertama kali di jalankan dan membutuhkan aksi *login*.

1. Perintah umum dalam **membuat file seeder** adalah seperti berikut

```
php artisan make:seeder <nama-class-seeder>
```

Perintah tersebut akan men-generate file seeder pada folder `PWL_POS/database/seeder`

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

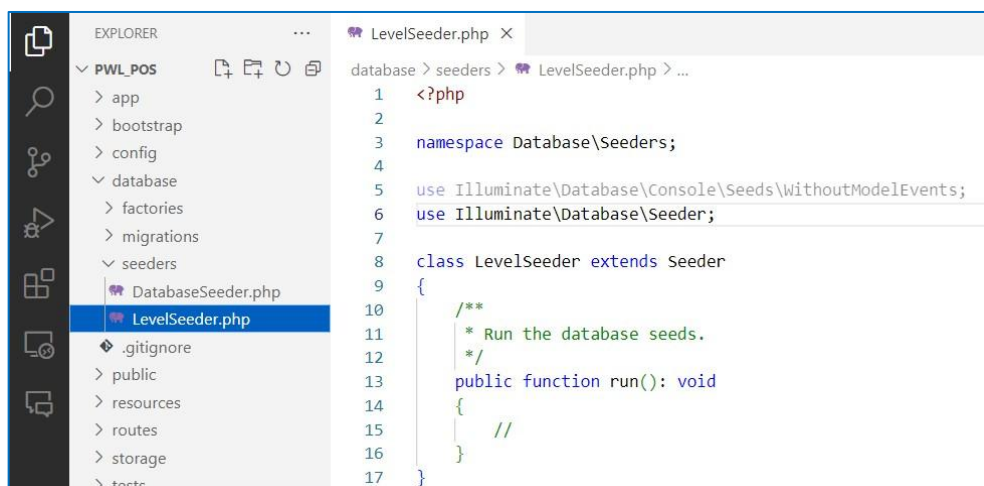
```
php artisan db:seed --class=<nama-class-seeder>
```

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.

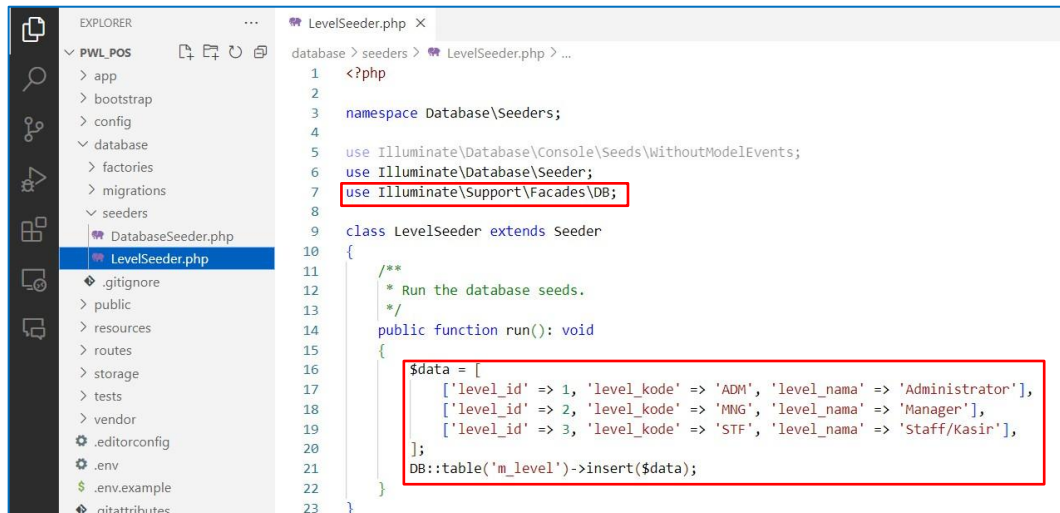
### Praktikum 3 – Membuat file *seeder*

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

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

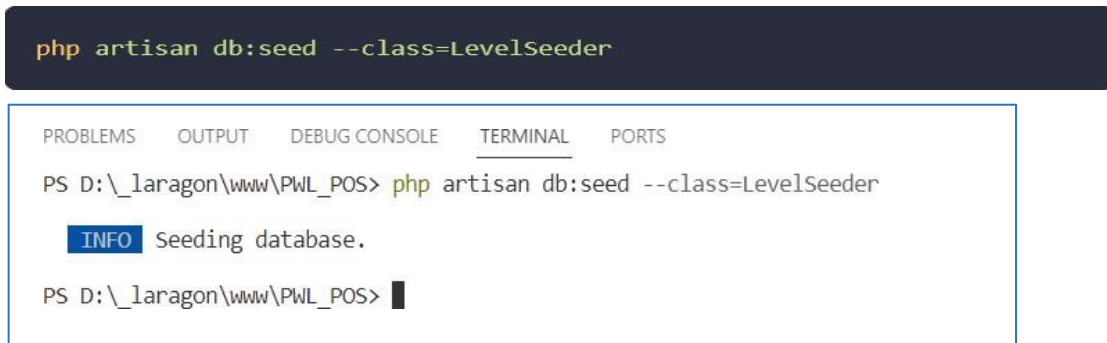


2. Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function `run()`



```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```

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



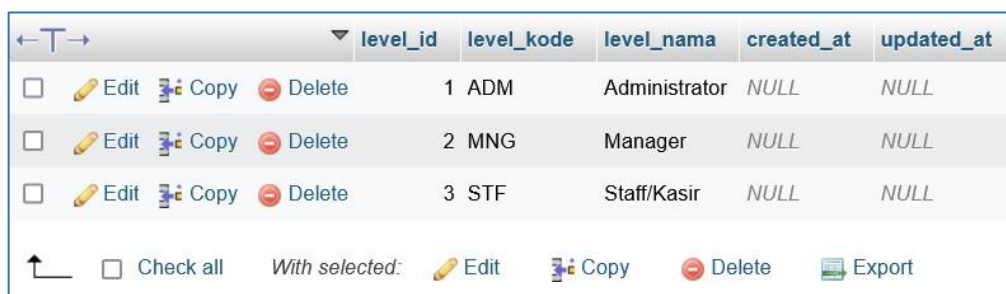
```
php artisan db:seed --class=LevelSeeder

PS D:\_laragon\www\PWL_POS> php artisan db:seed --class=LevelSeeder

INFO Seeding database.

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

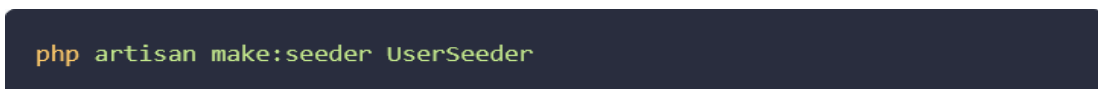
4. Ketika *seeder* berhasil dijalankan maka akan tampil data pada table `m_level`



	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> 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

☐ Check all With selected: Edit Copy Delete Export

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



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



6. Modifikasi file `class UserSeeder` seperti berikut

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

7. Jalankan perintah untuk mengeksekusi class `UserSeeder`

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

8. Perhatikan hasil seeder pada table `m_user`

	user_id	level_id	username	nama	password
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6fXV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhlFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...
↑ <input type="checkbox"/> Check all With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export					

9. Ok, data seeder berhasil di masukkan ke database.
10. Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut

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

**Hasil :** saya melakukan sesuai dengan apa yang sudah di contohkan pada praktikum sebelumnya untuk pengisian (seed) ini. Berikut langkah-langkah yang saya lakukan:



## a) M\_kategori

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class M_kategoriSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data = [
14             ['kategori_id' => 1, 'kategori_kode' => 'ELC', 'kategori_nama' =>
15                 'Elektronik'],
16             ['kategori_id' => 2, 'kategori_kode' => 'FNB', 'kategori_nama' =>
17                 'Makanan & Minuman'],
18             ['kategori_id' => 3, 'kategori_kode' => 'FAS', 'kategori_nama' =>
19                 'Fashion'],
20             ['kategori_id' => 4, 'kategori_kode' => 'OTM', 'kategori_nama' =>
21                 'Otomotif'],
22             ['kategori_id' => 5, 'kategori_kode' => 'KSM', 'kategori_nama' =>
23                 'Kosmetik'],
24         ];
25         DB::table('m_kategori')->insert($data);
26     }
27 }
```

			id	created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>	Edit	Copy	Delete	1	NULL	NULL	1 ELC	Elektronik
<input type="checkbox"/>	Edit	Copy	Delete	2	NULL	NULL	2 FNB	Makanan & Minuman
<input type="checkbox"/>	Edit	Copy	Delete	3	NULL	NULL	3 FAS	Fashion
<input type="checkbox"/>	Edit	Copy	Delete	4	NULL	NULL	4 OTM	Otomotif
<input type="checkbox"/>	Edit	Copy	Delete	5	NULL	NULL	5 KSM	Kosmetik

## b) M\_barang

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Seeder;
6 use Illuminate\Support\Facades\DB;
7
8 class MBarangSeeder extends Seeder
9 {
10     public function run(): void
11     {
12         $data = [
13             ['barang_id' => 1, 'kategori_id' => 1, 'barang_kode' => 'LAP001', 'l
14             ['barang_id' => 2, 'kategori_id' => 1, 'barang_kode' => 'SMP002', 'l
15             ['barang_id' => 3, 'kategori_id' => 2, 'barang_kode' => 'RT003', 'ba
16             ['barang_id' => 4, 'kategori_id' => 2, 'barang_kode' => 'SS004', 'ba
17             ['barang_id' => 5, 'kategori_id' => 3, 'barang_kode' => 'KS005', 'ba
18             ['barang_id' => 6, 'kategori_id' => 3, 'barang_kode' => 'SP006', 'ba
19             ['barang_id' => 7, 'kategori_id' => 4, 'barang_kode' => 'BM007', 'ba
20             ['barang_id' => 8, 'kategori_id' => 4, 'barang_kode' => 'OL008', 'ba
21             ['barang_id' => 9, 'kategori_id' => 5, 'barang_kode' => 'LP009', 'ba
22             ['barang_id' => 10, 'kategori_id' => 5, 'barang_kode' => 'BD010', 'l
23         ];
24
25         DB::table('m_barang')->insert($data);
26     }
27 }
```

	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	LAP001	Laptop	500000	550000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	1	SMP002	Smartphone	300000	350000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2	RT003	Roti	5000	7000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	2	SS004	Susu	15000	20000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	3	KS005	Kaos	50000	75000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	3	SP006	Sepatu	200000	250000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	4	BM007	Ban Motor	150000	200000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	4	OL008	Oli	50000	75000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	5	LP009	Lipstik	20000	35000	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	5	BD010	Bedak	30000	50000	NULL	NULL

### c) T\_stok

```
<?php

namespace Database\Seeders;

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

class TstokSeeder extends Seeder
{
    public function run(): void
    {
        $data = [
            ['barang_id' => 1, 'user_id' => 1, 'stok_tanggal' => Carbon::now(), 'stok_jumlah' => 50],
            ['barang_id' => 2, 'user_id' => 2, 'stok_tanggal' => Carbon::now()->subDays(1), 'stok_jumlah' => 30],
            ['barang_id' => 3, 'user_id' => 3, 'stok_tanggal' => Carbon::now()->subDays(2), 'stok_jumlah' => 100],
            ['barang_id' => 4, 'user_id' => 1, 'stok_tanggal' => Carbon::now()->subDays(3), 'stok_jumlah' => 20],
            ['barang_id' => 5, 'user_id' => 2, 'stok_tanggal' => Carbon::now()->subDays(4), 'stok_jumlah' => 75],
            ['barang_id' => 6, 'user_id' => 3, 'stok_tanggal' => Carbon::now()->subDays(5), 'stok_jumlah' => 40],
            ['barang_id' => 7, 'user_id' => 1, 'stok_tanggal' => Carbon::now()->subDays(6), 'stok_jumlah' => 10],
            ['barang_id' => 8, 'user_id' => 2, 'stok_tanggal' => Carbon::now()->subDays(7), 'stok_jumlah' => 25],
            ['barang_id' => 9, 'user_id' => 3, 'stok_tanggal' => Carbon::now()->subDays(8), 'stok_jumlah' => 60],
            ['barang_id' => 10, 'user_id' => 1, 'stok_tanggal' => Carbon::now()->subDays(9), 'stok_jumlah' => 15],
        ];

        DB::table('t_stok')->insert($data); //barang_id dan user_id mengacu pada data yang sudah ada
    }
}
```

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	1	2025-03-03 06:42:46	50	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	2	2025-03-02 06:42:46	30	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	3	2025-03-01 06:42:46	100	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	4	1	2025-02-28 06:42:46	20	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	5	2	2025-02-27 06:42:46	75	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	6	3	2025-02-26 06:42:46	40	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	7	1	2025-02-25 06:42:46	10	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	8	2	2025-02-24 06:42:46	25	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	9	3	2025-02-23 06:42:46	60	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	10	1	2025-02-22 06:42:46	15	NULL	NULL

#### d) T\_penjualan

```
<?php

namespace Database\Seeders;

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

Codeium: Refactor | Explain
class TPenjualanSeeder extends Seeder
{
    Codeium: Refactor | Explain | Generate Function Comment | X
    public function run(): void
    {
        $data = [
            ['user_id' => 1, 'pembeli' => 'Andi', 'penjualan_kode' => 'TRX001',
            ['user_id' => 2, 'pembeli' => 'Budi', 'penjualan_kode' => 'TRX002',
            ['user_id' => 3, 'pembeli' => 'Citra', 'penjualan_kode' => 'TRX003',
            ['user_id' => 1, 'pembeli' => 'Dina', 'penjualan_kode' => 'TRX004',
            ['user_id' => 2, 'pembeli' => 'Eko', 'penjualan_kode' => 'TRX005',
            ['user_id' => 3, 'pembeli' => 'Faisal', 'penjualan_kode' => 'TRX006',
            ['user_id' => 1, 'pembeli' => 'Gita', 'penjualan_kode' => 'TRX007',
            ['user_id' => 2, 'pembeli' => 'Hadi', 'penjualan_kode' => 'TRX008',
            ['user_id' => 3, 'pembeli' => 'Indah', 'penjualan_kode' => 'TRX009',
            ['user_id' => 1, 'pembeli' => 'Joko', 'penjualan_kode' => 'TRX010',
        ];

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

	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	1	1	Andi	TRX001	2025-03-03 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	2	2	Budi	TRX002	2025-03-02 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	3	3	Citra	TRX003	2025-03-01 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	4	1	Dina	TRX004	2025-02-28 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	5	2	Eko	TRX005	2025-02-27 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	6	3	Faisal	TRX006	2025-02-26 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	7	1	Gita	TRX007	2025-02-25 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	8	2	Hadi	TRX008	2025-02-24 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	9	3	Indah	TRX009	2025-02-23 06:53:46	NULL	NULL
<input type="checkbox"/> Edit <input type="copy"/> <input type="delete"/>	10	1	Joko	TRX010	2025-02-22 06:53:46	NULL	NULL

#### e) T\_penjualan\_detail

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Seeder;
6 use Illuminate\Support\Facades\DB;
7 use Carbon\Carbon;
8
9 Codeium: Refactor | Explain
10 class TPenjualanDetailSeeder extends Seeder
11 {
12     Codeium: Refactor | Explain | Generate Function Comment | X
13     public function run(): void
14     {
15         $data = [
16             ['detail_id' => 1, 'penjualan_id' => 1, 'barang_id' => 1, 'harga' => 1,
17             ['detail_id' => 2, 'penjualan_id' => 1, 'barang_id' => 2, 'harga' => 2,
18             ['detail_id' => 3, 'penjualan_id' => 2, 'barang_id' => 3, 'harga' => 3,
19             ['detail_id' => 4, 'penjualan_id' => 2, 'barang_id' => 4, 'harga' => 4,
20             ['detail_id' => 5, 'penjualan_id' => 3, 'barang_id' => 5, 'harga' => 5,
21             ['detail_id' => 6, 'penjualan_id' => 4, 'barang_id' => 6, 'harga' => 6,
22             ['detail_id' => 7, 'penjualan_id' => 5, 'barang_id' => 7, 'harga' => 7,
23             ['detail_id' => 8, 'penjualan_id' => 6, 'barang_id' => 8, 'harga' => 8,
24             ['detail_id' => 9, 'penjualan_id' => 7, 'barang_id' => 9, 'harga' => 9,
25             ['detail_id' => 10, 'penjualan_id' => 8, 'barang_id' => 10, 'harga' => 10,
26         ];
27
28         DB::table('t_penjualan_detail')->insert($data);
29     }
30 }
```

detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
1	1	1	8000000	1	2025-03-03 06:58:11	2025-03-03 06:58:11
2	1	2	5000000	1	2025-03-03 06:58:11	2025-03-03 06:58:11
3	2	3	20000	3	2025-03-03 06:58:11	2025-03-03 06:58:11
4	2	4	15000	2	2025-03-03 06:58:11	2025-03-03 06:58:11
5	3	5	100000	2	2025-03-03 06:58:11	2025-03-03 06:58:11
6	4	6	300000	1	2025-03-03 06:58:11	2025-03-03 06:58:11
7	5	7	250000	4	2025-03-03 06:58:11	2025-03-03 06:58:11
8	6	8	50000	5	2025-03-03 06:58:11	2025-03-03 06:58:11
9	7	9	75000	2	2025-03-03 06:58:11	2025-03-03 06:58:11
10	8	10	120000	3	2025-03-03 06:58:11	2025-03-03 06:58:11

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

## D. DB FACADE

DB Façade merupakan fitur dari Laravel yang digunakan untuk melakukan *query* secara langsung dengan mengetikkan perintah SQL secara utuh (*raw query*). Disebut *raw query* (query mentah) karena penulisan query pada DB Façade langsung ditulis sebagaimana yang biasa dituliskan pada database, seperti “`select * from m_user`” atau “`insert into m_user...`” atau “`update m_user set ... Where ...`”

*Raw query* adalah cara paling dasar dan tradisional yang ada di Laravel. Raw query terasa familiar karena biasa kita pakai ketika melakukan query langsung ke database.

### INFO

Dokumentasi penggunaan DB Façade bisa dicek di laman ini  
<https://laravel.com/docs/10.x/database#running-queries>

Terdapat banyak method yang bisa digunakan pada DB Façade ini. Akan tetapi yang kita pelajari cukup 4 (empat) method yang umum dipakai, yaitu

a. `DB::select()`

Method ini digunakan untuk mengambil data dari database. Method ini **mengembalikan** (*return*) data hasil *query*. Contoh

```
DB::select('select * from m_user'); //Query semua data pada tabel m_user
```

```
DB::select('select * from m_user where level_id = ?', [1]); //Query tabel m_user dengan level_id = 1
```

```
DB::select('select * from m_user where level_id = ? and username = ?', [1, 'admin']);
```

b. `DB::insert()`

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

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

c. `DB::update()`

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

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

d. `DB::delete()`

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

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

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

## Praktikum 4 – Implementasi DB Facade

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

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

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

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

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

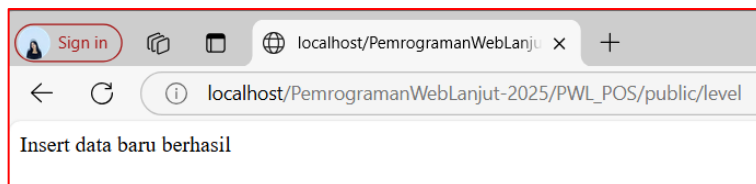
```
LevelController.php X  web.php
app > Http > Controllers > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12 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



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

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

### Hasil :



				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	4	CUS	Pelanggan	2025-03-03 07:16:40	NULL

- a) User mengakses URL `/level` di browser → Laravel memproses request.
  - b) Route menangkap request → Route `/level` memanggil `LevelController@index`.
  - c) Controller mengeksekusi logika bisnis → Bisa jadi ada perintah insert ke tabel pelanggan (`m_pelanggan`).
  - d) Database menyimpan data baru → Pelanggan dengan kode CUS + timestamp dimasukkan ke database bersama waktu (`created_at`).
  - e) Response dikirim ke browser → Bisa berupa tampilan data `m_level` atau pesan sukses karena ada `DB::insert`
5. Selanjutnya, kita modifikasi lagi file `LevelController` untuk meng-*update* data di table `m_level` seperti berikut

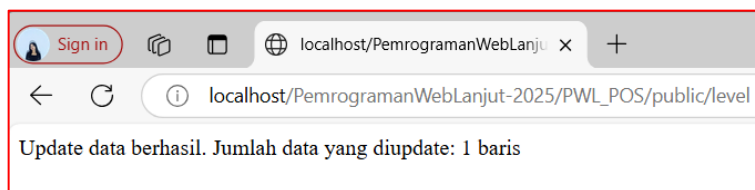
```

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

```

6. Kita coba jalankan di browser dengan url [localhost/PWL\\_POS/public/level](localhost/PWL_POS/public/level) lagi dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`

**Hasil :**



- Tidak ada row baru yang muncul karena kode yang dieksekusi adalah update, bukan insert.
- `DB::update` hanya mengubah nilai pada row yang sudah ada, bukan menambah baris baru.
- Jika ingin menambahkan data baru, gunakan `DB::insert`.
- Gunakan pengecekan `exists()` agar tidak terjadi duplikasi data

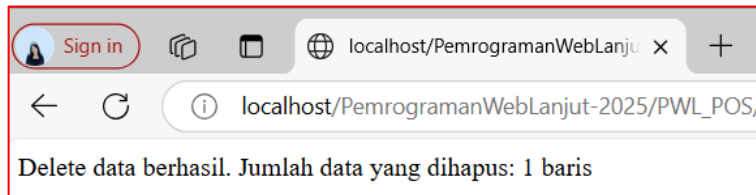
7. Kita coba modifikasi lagi file `LevelController` untuk melakukan proses hapus data

```

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 }

```

## Hasil :



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

a) Ini akan menghapus baris dalam tabel `m_level` yang memiliki `level_kode = 'CUS'`.

b) Jika ada baris yang cocok dengan kondisi tersebut, maka akan dihapus.

c) \$row akan menyimpan jumlah baris yang terhapus

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

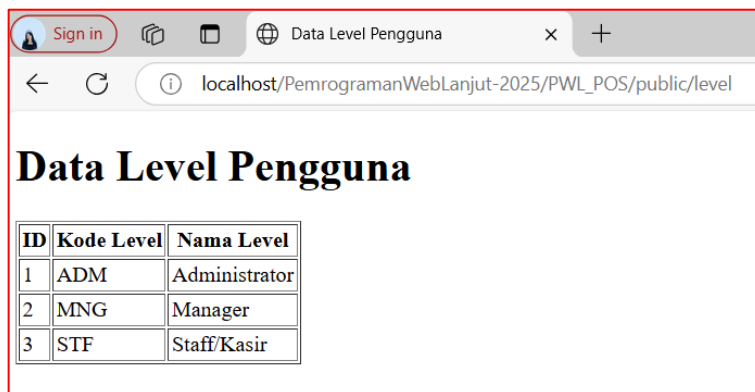
```
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS',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

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

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

10. Silahkan dicoba pada browser dan amati apa yang terjadi

**Hasil :** ketika akses URL laravel cari route yg sesuai, kemudian laravel mengecek apakah route ada yang mengarah ke levelcontrol@index. Lalu Laravel menjalankan query SQL menggunakan DB::select() untuk mengambil data dari tabel m\_level. Data yang diambil disimpan dalam variabel \$data. Setelah dapat data dari DB laravel memanggil return view kemudian mengirimkan data ke level.blade.php kemudian di file level.blade.php menampilkan data menggunakan foreach.



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 Facade yang mengharuskan kita menulis perintah SQL, pada *query builder* perintah SQL ini diakses menggunakan method. Jadi, kita tidak menulis perintah SQL secara langsung, melainkan cukup memanggil method-method yang ada di *query builder*.

Query builder membuat kode kita menjadi rapi dan lebih mudah dibaca. Selain itu *query builder* tidak terikat ke satu jenis database, jadi query builder bisa digunakan untuk mengakses berbagai jenis database seperti MySQL, MariaDB, PostgreSQL, SQL Server, dll. Jika suatu saat ingin beralih dari database MySQL ke PostgreSQL, tidak akan banyak kendala. Namun kelemahan dari *query builder* adalah kita harus mengetahui method-method apa saja yang ada di *query builder*.

### INFO

Dokumentasi penggunaan Query Builder pada Laravel bisa dicek di laman ini

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

Ciri khas *query builder* Laravel adalah kita tentukan dahulu target table yang akan kita akses untuk operasi CRUD.

```
DB::table('<nama-tabel>'); // query builder untuk melakukan operasi CRUD pada tabel yang dituju
```

Perintah pertama yang dilakukan pada query builder adalah menentukan nama table yang akan dilakukan operasi CRUD. Kemudian baru disusul method yang ingin digunakan sesuai dengan peruntukannya. Contoh

- Perintah untuk *insert* data dengan method `insert()`

```
DB::table('m_kategori')->insert(['kategori_kode' => 'SMP', 'kategori_nama' => 'Smartphone']);
```

Query yang dihasilkan dari kode di atas adalah

```
insert into m_kategori(kategori_kode, kategori_nama) values('SMP', 'Smartphone');
```

- Perintah untuk *update* data dengan method `where()` dan `update()`

```
DB::table('m_kategori')->where('kategori_id', 1)->update(['kategori_nama' => 'Makanan Ringan']);
```

Query yang dihasilkan dari kode di atas adalah

```
update m_kategori set kategori_nama = 'Makanan Ringan' where kategori_id = 1;
```

- c. Perintah untuk *delete* data dengan method `where()` dan `delete()`

```
DB::table('m_kategori')->where('kategori_id', 9) ->delete();
```

Query yang dihasilkan dari kode di atas adalah

```
delete from m_kategori where kategori_id = 9;
```

- d. Perintah untuk ambil data

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

## Praktikum 5 – Implementasi *Query Builder*

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

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

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

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

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



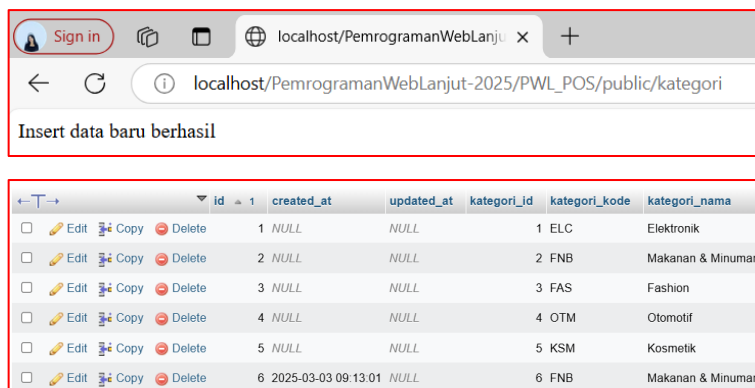
```

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

```

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

**Hasil : z**



Insert data baru berhasil

	id	created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>	1	NULL	NULL	1	ELC	Elektronik
<input type="checkbox"/>	2	NULL	NULL	2	FNB	Makanan & Minuman
<input type="checkbox"/>	3	NULL	NULL	3	FAS	Fashion
<input type="checkbox"/>	4	NULL	NULL	4	OTM	Otomotif
<input type="checkbox"/>	5	NULL	NULL	5	KSM	Kosmetik
<input type="checkbox"/>	6	2025-03-03 09:13:01	NULL	6	FNB	Makanan & Minuman

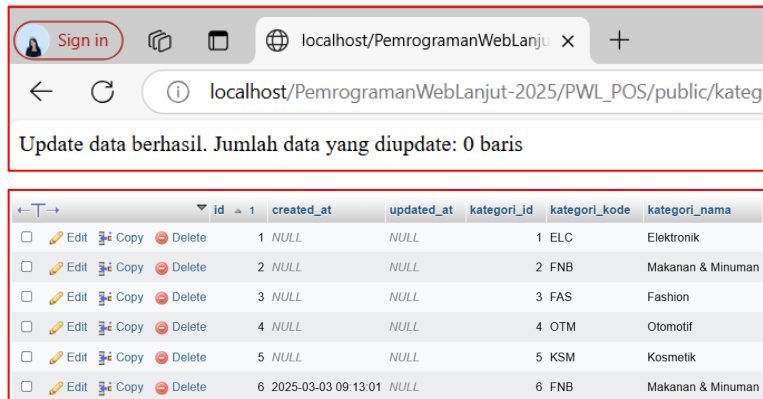
5. Selanjutnya, kita modifikasi lagi file `KategoriController` untuk meng-*update* data di table `m_kategori` seperti berikut

```

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 }

```

## Hasil :



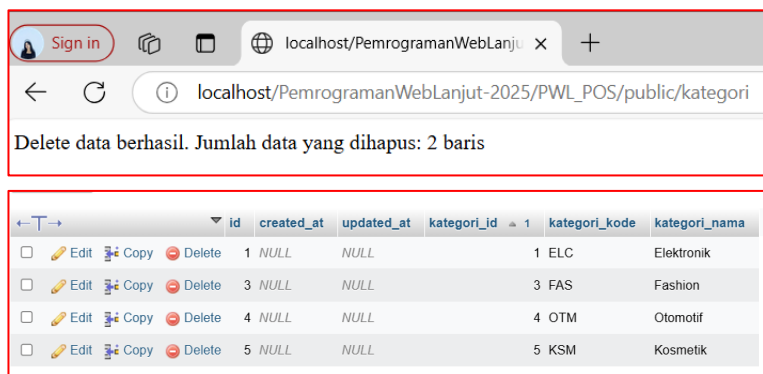
Update data berhasil. Jumlah data yang diupdate: 0 baris

					id	created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>	Edit	Copy	Delete		1	NULL	NULL	1	ELC	Elektronik
<input type="checkbox"/>	Edit	Copy	Delete		2	NULL	NULL	2	FNB	Makanan & Minuman
<input type="checkbox"/>	Edit	Copy	Delete		3	NULL	NULL	3	FAS	Fashion
<input type="checkbox"/>	Edit	Copy	Delete		4	NULL	NULL	4	OTM	Otomotif
<input type="checkbox"/>	Edit	Copy	Delete		5	NULL	NULL	5	KSM	Kosmetik
<input type="checkbox"/>	Edit	Copy	Delete		6	2025-03-03 09:13:01	NULL	6	FNB	Makanan & Minuman

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`

```
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 }
```

## Hasil :



Delete data berhasil. Jumlah data yang dihapus: 2 baris

					id	created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>	Edit	Copy	Delete		1	NULL	NULL	1	ELC	Elektronik
<input type="checkbox"/>	Edit	Copy	Delete		3	NULL	NULL	3	FAS	Fashion
<input type="checkbox"/>	Edit	Copy	Delete		4	NULL	NULL	4	OTM	Otomotif
<input type="checkbox"/>	Edit	Copy	Delete		5	NULL	NULL	5	KSM	Kosmetik

7. 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 }

```

8. 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>

```

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

### Hasil :

- Menggunakan Query Builder (`DB::table()`) untuk mengambil semua data dari tabel `m_kategori`.
- Hasilnya berupa Collection Laravel yang berisi semua baris data dari tabel tersebut.
- Menampilkan halaman `kategori.blade.php` yang ada di folder `resources/views/`.
- Mengirimkan data dari database ke dalam view `kategori`.
- data bisa digunakan di dalam Blade sebagai variabel (`$data`).



The screenshot shows a web browser window with the title 'Data Kategori Barang'. The address bar displays 'localhost/PemrogramanWebLanjut-2025/PWL\_POS/public/kategori'. Below the title, there is a table with three columns: 'ID', 'Kode Kategori', and 'Nama Kategori'. The table contains five rows of data.

ID	Kode Kategori	Nama Kategori
1	ELC	Elektronik
3	FAS	Fashion
4	OTM	Otomotif
5	KSM	Kosmetik

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

## F. ELOQUENT ORM

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

### INFO

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

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

Perintah untuk membuat model adalah sebagai berikut

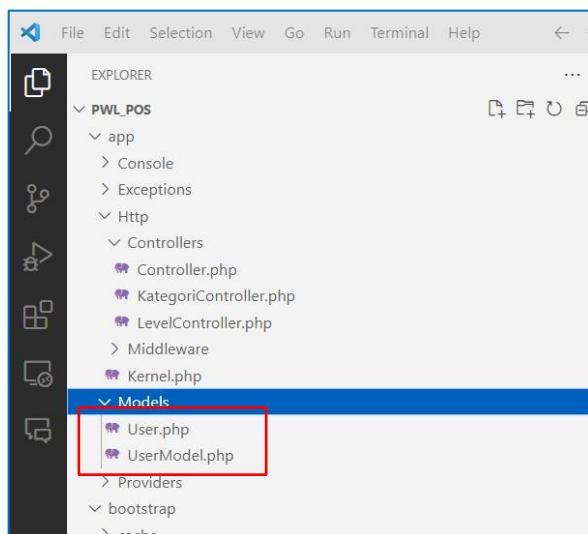
```
php artisan make:model <nama-model-CamelCase>
```

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

## Praktikum 6 – Implementasi Eloquent ORM

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

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



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

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

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

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

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

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

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



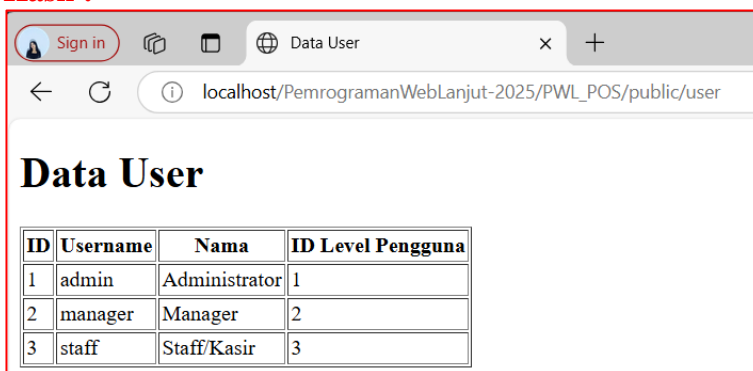
```

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

```

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

**Hasil :**



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

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

```

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

```

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

## Hasil :

Error ini muncul karena ada foreign key constraint pada `m_user.level_id`, yang harus merujuk ke `m_level.level_id`.

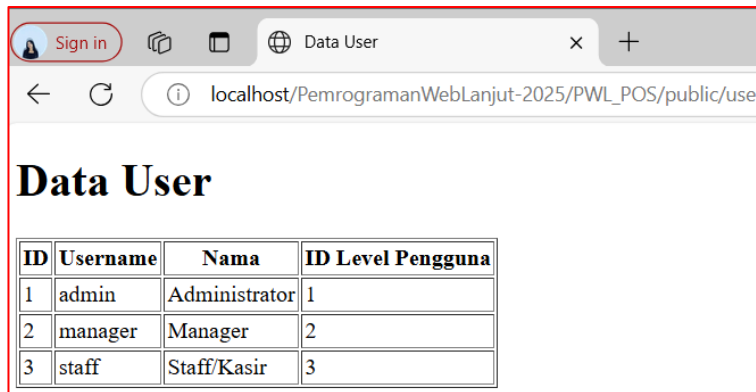


10. Kita modifikasi lagi file `UserController` menjadi seperti berikut

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

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

**Hasil :**



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

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

## G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari **APP\_KEY** pada *file setting .env* Laravel?

**Jawaban :** APP\_KEY digunakan untuk mengenkripsi data dalam aplikasi Laravel, seperti session dan password. Tanpa APP\_KEY yang valid, fitur keamanan Laravel tidak akan bekerja dengan optimal

2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk **APP\_KEY**?

**Jawaban :** Cukup jalankan perintah berikut di terminal saat ada di root proyek Laravel:  
**php artisan key:generate** , yang tujuannya menghasilkan key baru dan otomatis disimpan dalam file *.env*

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

**Jawaban :**

Secara default, Laravel punya tiga file migrasi:

- a) `create_users_table.php` → Membuat tabel pengguna
- b) `create_password_resets_table.php` → Untuk menyimpan token reset password
- c) `create_failed_jobs_table.php` → Menyimpan informasi job yang gagal dieksekusi

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

**Jawaban :**

Ini otomatis menambahkan dua kolom:

- 1) `created_at` → Waktu saat data dibuat
- 2) `updated_at` → Waktu saat data terakhir diperbarui

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

**Jawaban :** Ini menghasilkan BIGINT (Unsigned Big Integer) dengan auto-increment.

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

**Jawaban :** ada perbedaan khusus,

- a) `$table->id()`; → Membuat kolom id sebagai primary key default.
- b) `$table->id('level_id')`; → Sama seperti di atas, tapi nama kolomnya `level_id` bukan `id`.

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

**Jawaban :** Ini memastikan data dalam kolom tersebut tidak ada yang duplikat atau sama. Biasanya dipakai untuk email, username, atau kode unik lainnya.

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')` ?

**Jawaban :** Karena `level_id` di `m_user` adalah foreign key yang merujuk ke `level_id` di `m_level`. Sedangkan `m_level` pakai `$table->id('level_id')`, yang secara default adalah `unsignedBigInteger`

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

**Jawaban :**

- a) Hash digunakan untuk mengenkripsi password sebelum disimpan ke database.
- b) `Hash::make('1234')`; berarti mengenkripsi string '1234' supaya nggak disimpan dalam bentuk teks biasa di database.

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

**Jawaban :** Ini disebut binding parameter. Tujuannya untuk mencegah SQL Injection dan membuat query lebih aman/safe

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

**Jawaban :**

- a) `$table = 'm_user';` → Memberitahu Laravel bahwa model ini pakai tabel `m_user`.
- b) `$primaryKey = 'user_id';` → Mengubah primary key default (`id`) menjadi `user_id`.

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

**Jawaban :** Sebenarnya depends, tapi kalau menurut saya pribadi paling mudah adalah Eloquent ORM, alasannya adalah :

- 1) DB Façade: Cocok untuk query mentah atau kompleks yang nggak bisa di-handle ORM.
- 2) Query Builder: Fleksibel dan lebih cepat daripada ORM karena langsung berinteraksi dengan database.
- 3) Eloquent ORM: Paling mudah karena berbasis objek, lebih readable, dan otomatis meng-handle relasi antar tabel.

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