



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**.

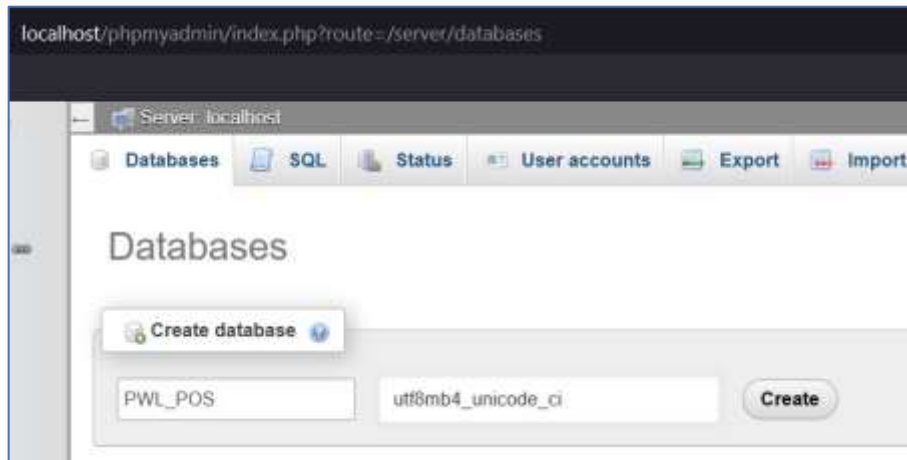
Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. PENGATURAN DATABASE

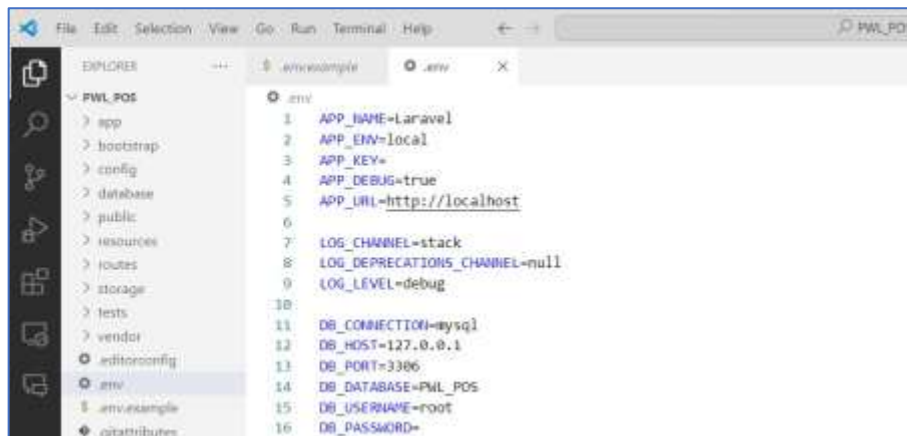
Database atau basis data menjadi komponen penting dalam membangun sistem. Hal ini dikarenakan database menjadi tempat untuk menyimpan data-data transaksi yang ada pada sistem. Koneksi ke database perlu kita atur agar sesuai dengan database yang kita gunakan.

Praktikum 1 - pengaturan database:

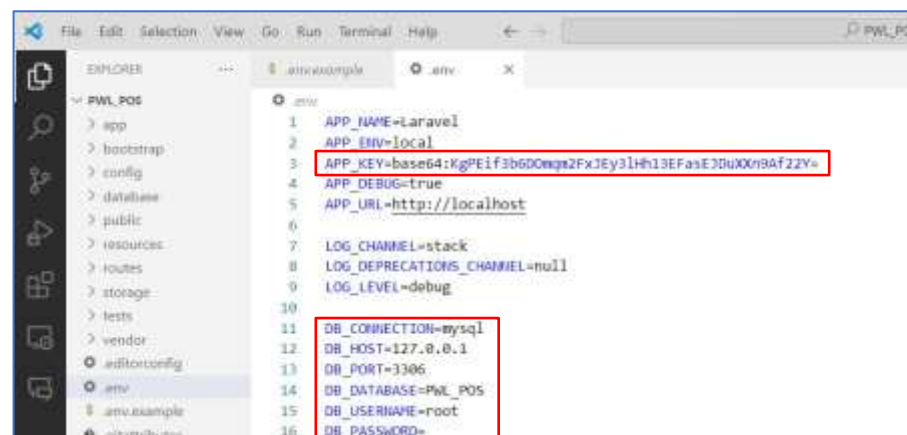
1. Buka aplikasi phpMyAdmin, dan buat database baru dengan nama **PWL_POS**



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*.



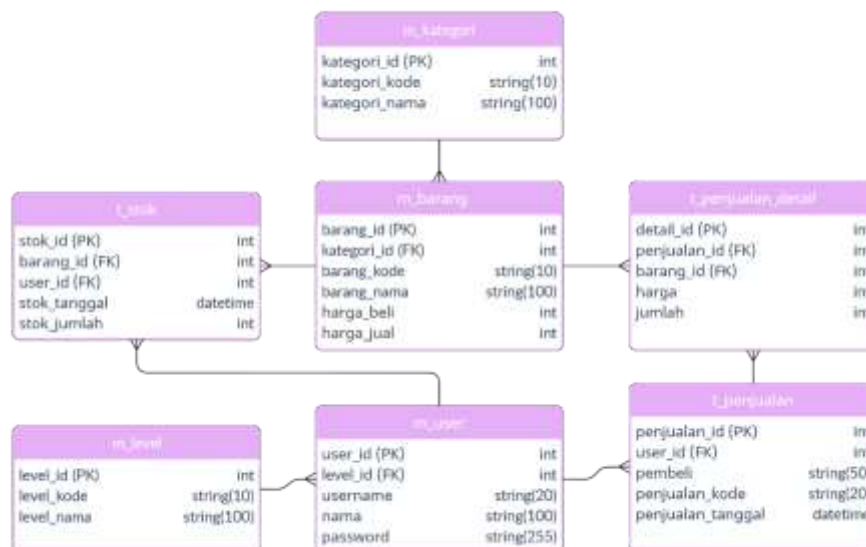
B. MIGRATION

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*

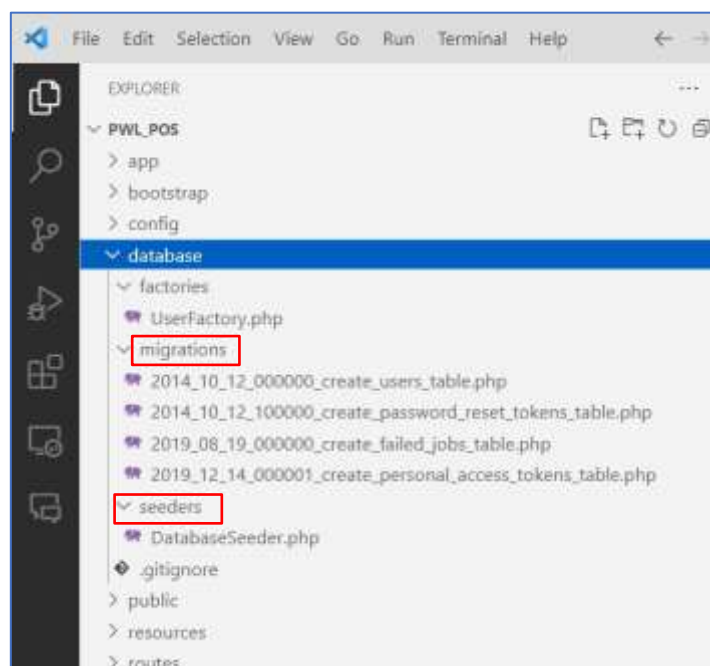
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- 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.

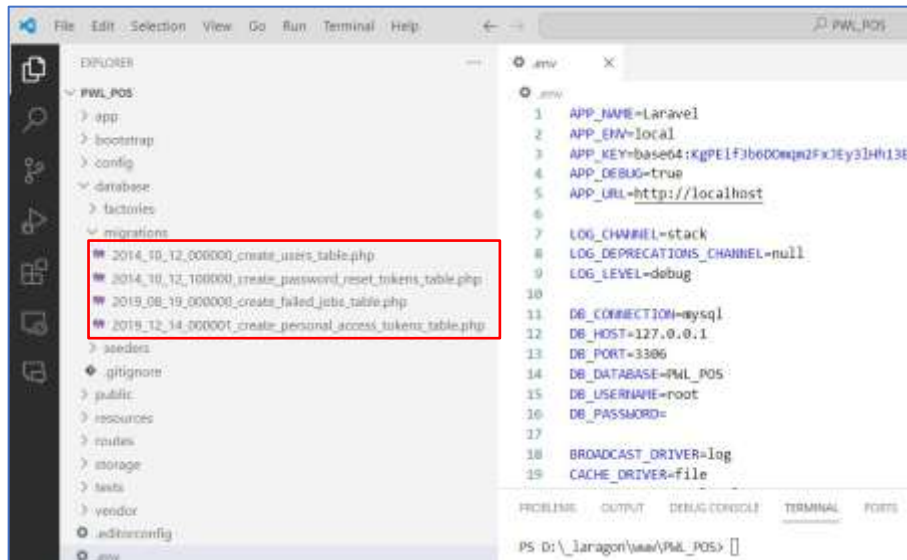
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder **PWL_POS/database**





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

```
1 return new class extends Migration {
2     {
3         /**
4          * Run the migrations.
5          */
6         public function up(): void
7         {
8             Schema::create('m_level', function (Blueprint $table) {
9                 $table->id('level_id');
10                $table->string('level_kode', 10)->unique();
11                $table->string('level_nama', 100);
12                $table->timestamps();
13            });
14        }
15
16        /**
17         * Reverse the migrations.
18         */
19        public function down(): void
20        {
21            Schema::dropIfExists('m_level');
22        }
23    }
24 }
```

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

INFO

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

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

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

```
PS C:\laragon\www\PM_2025\PM_P05> php artisan migrate
[INFO] Preparing database.
Creating migration table ..... 32ms DONE
[INFO] Running migrations.
2025-01-05 00:00:00 create users_table ..... 30ms DONE
2025-01-05 00:00:00 create password reset tokens_table ..... 10ms DONE
2025-01-05 00:00:00 create failed jobs_table ..... 47ms DONE
2025-01-05 00:00:00 create personal access tokens_table ..... 37ms DONE
2025-01-05 00:00:00 create m_level_table ..... 23ms DONE
PS C:\laragon\www\PM_2025\PM_P05>
```

```
PS C:\laragon\www\PM_2025\PM_P05> php artisan migrate
[INFO] Preparing database.
Creating migration table ..... 455ms DONE
[INFO] Running migrations.
2025-01-05 00:00:00 create users_table ..... 96ms DONE
2025-01-05 00:00:00 create password reset tokens_table ..... 13ms DONE
2025-01-05 00:00:00 create failed jobs_table ..... 56ms DONE
2025-01-05 00:00:00 create personal access tokens_table ..... 30ms DONE
2025-01-05 00:00:00 create m_level_table ..... 20ms DONE
```




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

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

<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0 InnoDB utf8mb4_unicode_ci 16.0 KiB
----------------------------------	---	--------------------------------------

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

<input type="checkbox"/> m_kategori	★ Browse Structure Search Insert Empty Drop	0 InnoDB utf8mb4_unicode_ci 16.0 KiB
-------------------------------------	---	--------------------------------------

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

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

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

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

```
PS C:\laragon\www\PM2025\pwl_week3\PM2POS> php artisan make:migration create_m_user_table --table=m_user
[INFO] Migration [C:\laragon\www\PM2025\pwl_week3\PM2POS\database\migrations\2025_03_08_081153_create_m_user_table.php] created successfully.
```

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

```
11  namespace Database\Migrations;
12
13  use Illuminate\Database\Schema\Blueprint;
14  use Illuminate\Support\Facades\Schema;
15
16  /**
17   * Create a new migration file.
18   *
19   * @param string $table
20   * @return void
21   */
22  public function up(): void
23  {
24      Schema::create($table, function (Blueprint $table) {
25          $table->id('user_id');
26          $table->unsignedBigInteger('level_id')->index();
27          $table->string('username', 20)->unique();
28          $table->string('nama', 100);
29          $table->string('password');
30          $table->timestamps();
31
32          $table->foreign('level_id')->references('level_id')->on('m_level');
33      });
34  }
35
36  /**
37   * Reverse the migrations.
38   *
39   * @return void
40   */
41  public function down(): void
42  {
43      Schema::dropIfExists($table);
44  }
45  }
```



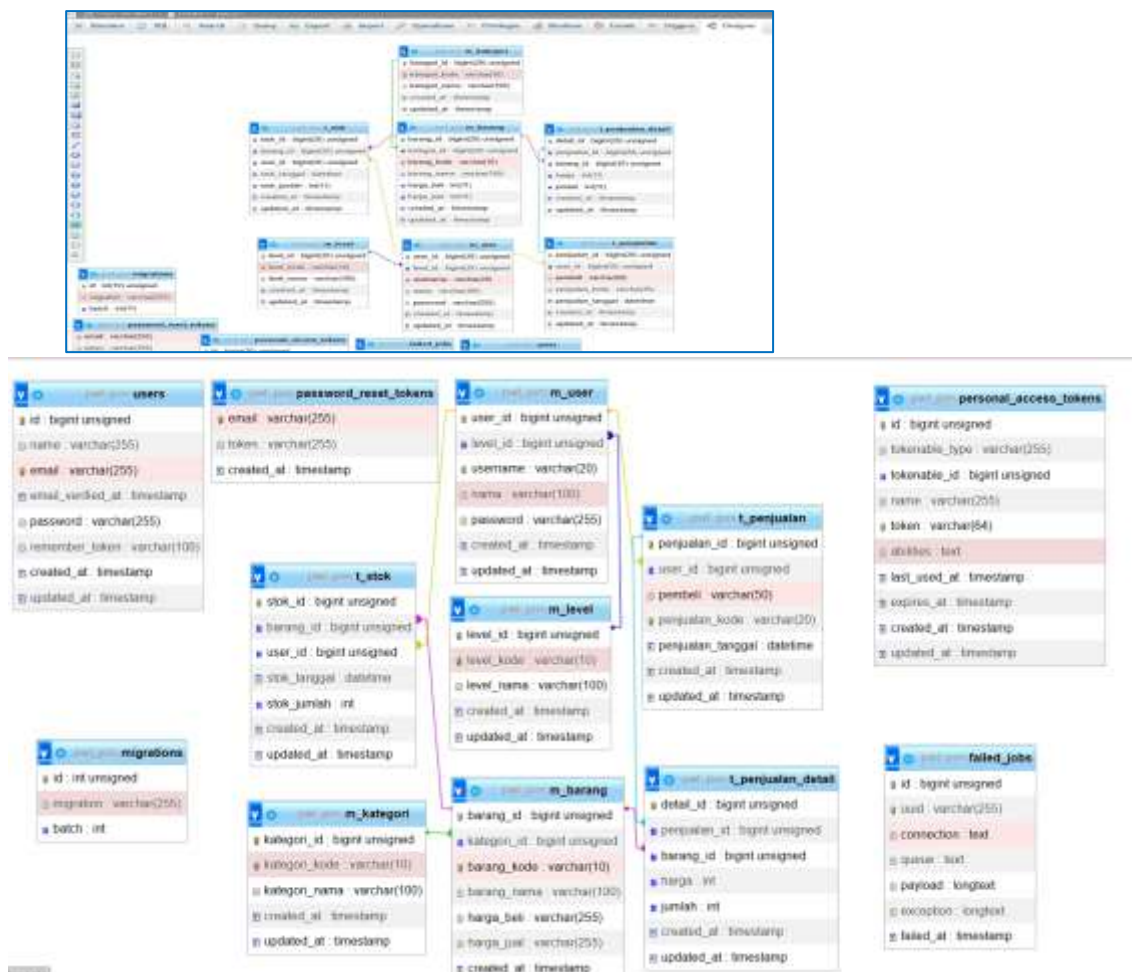
3. Simpan kode program Langkah 2, dan jalankan perintah **php artisan migrate**. Amati apa yang terjadi pada database.



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

m_barang
t_penjualan
t_stok
t_penjualan_detail

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



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



C. SEEDER

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

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

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

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

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

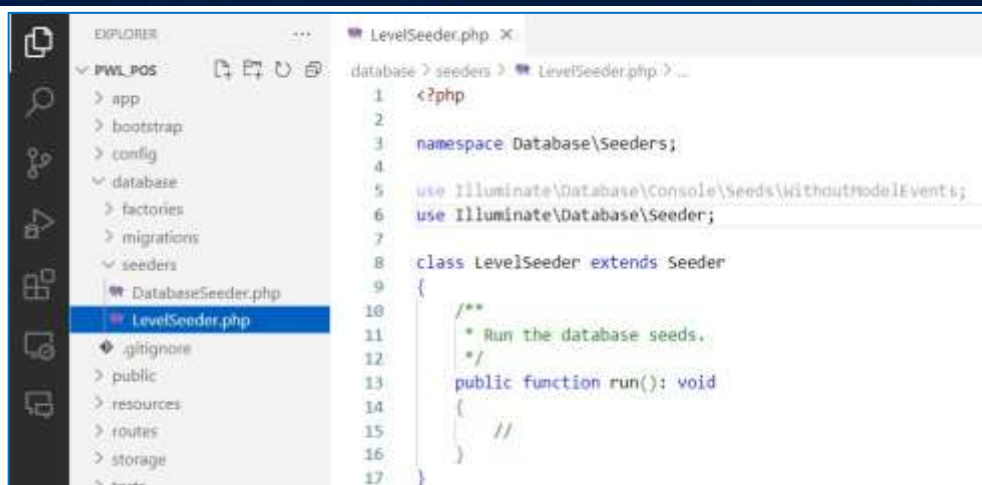
```
php artisan db:seed --class=<nama-class-seeder>
```

Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau *dummy* data untuk memudahkan pengujian dan pengembangan aplikasi kita. Sehingga fitur *seeder* bisa kita pakai dalam membuat sebuah aplikasi web.

Praktikum 3 – Membuat file *seeder*

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

```
PS C:\laragon\www\PWL2025\pwl_week3\PWLPOS> php artisan make:seeder LevelSeeder  
INFO Seeder [C:\laragon\www\PWL2025\pwl_week3\PWLPOS\database\seeder\LevelSeeder.php] created successfully.
```





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

```
PS C:\laragon\www\PWL2025\pwl_week3\PWL_POS> php artisan db:seed --class=LevelSeeder
INFO Seeding database.

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`

```
PS C:\laragon\www\PWL2025\pwl_week3\PWLPOS> php artisan make:seeder UserSeeder
INFO Seeder [C:\laragon\www\PWL2025\pwl_week3\PWLPOS\database\seeders\UserSeeder.php] created successfully.
PS C:\laragon\www\PWL2025\pwl_week3\PWLPOS> php artisan db:seed --class=UserSeeder
INFO Seeding database.
```

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\$Tevu4dD01CUAQpeM6H Vp LySwY 4oAKU7FzwS80XV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Aqns20FdPTeUgghz31muEhIFarUdxh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gz23TqGclW5pYeR0VL4o5OxPwb3Osk99VMYlBHnbJ9W...
<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

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

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

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

6. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/level lagi dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`
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. 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
11. Laporkan hasil Praktikum-4 ini dan *commit* perubahan pada *git*.
Perubahan pada no 3

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	STF	StaffKasir	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	CUS	Pelanggan	2025-03-05 12:44:41	NULL

perubahan pada no 5

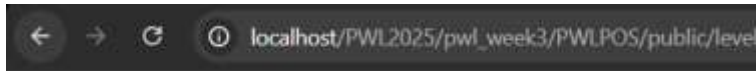
	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	STF	StaffKasir	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	CUS	Customer	2025-03-05 12:44:41	NULL



Perubahan pada no 7

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

perubahan pada no 9



Data Level Pengguna

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



E. QUERY BUILDER

Query builder adalah fitur yang disediakan Laravel untuk melakukan proses CRUD (*create, retrieve/read, update, delete*) pada database. Berbeda dengan *raw query* pada DB Facede yang mengharuskan kita menulis perintah SQL, pada *query builder* perintah SQL ini diakses menggunakan method. Jadi, kita tidak menulis perintah SQL secara langsung, melainkan cukup memanggil method-method yang ada di *query builder*.

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

INFO

Dokumentasi penggunaan Query Builder pada Laravel bisa dicek di laman ini

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

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

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

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

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

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

Query yang dihasilkan dari kode di atas adalah

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

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

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

Query yang dihasilkan dari kode di atas adalah

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



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

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

Query yang dihasilkan dari kode di atas adalah

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

- d. Perintah untuk ambil data

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

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

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

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ATM	Alat Makan	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	PRK	Perkakas	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	MNM	Minuman	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	PKB	Perlengkapan Bengkel	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	TKN	Teknologi	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	SNK	Snack/Makanan Ringan	2025-03-05 13:12:07	NULL

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

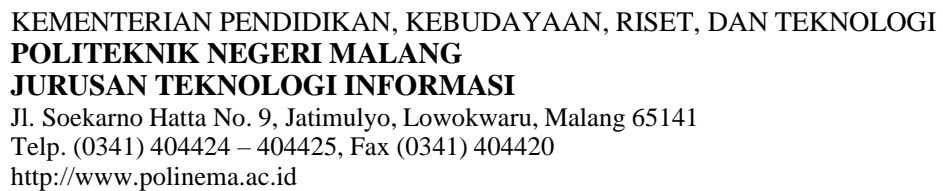
```
app > Http > Controllers > KategoriController.php > KategoriController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class KategoriController extends Controller
9 {
10     public function index()
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil';
19     }
20
21     public function update(Request $request)
22     {
23         $data = [
24             'kategori_kode' => $request->kategori_kode,
25             'kategori_nama' => $request->kategori_nama,
26             'updated_at' => now()
27         ];
28         DB::table('m_kategori')->where('kategori_kode', $request->kategori_kode)->update($data);
29         return 'Update data berhasil. Isilah data yang diupdate.';
30     }
31 }
```
















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

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ATM	Alat Makan	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	PRK	Perkakas	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	MNM	Minuman	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	PKB	Perlengkapan Bengkel	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	TKN	Teknologi	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	SNK	Camilan	2025-03-05 13:12:07	NULL

yang ada pada table `m_kategori`

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



		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	 Edit  Copy  Delete	1	ATM	Alat Makan	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	2	PRK	Perkakas	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	3	MNM	Minuman	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	4	PKB	Perlengkapan Bengkel	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	5	TKN	Teknologi	NULL	NULL

- ```

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' => 'Canilan']);
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 }
















```

- ```

1 // @author : @KunalKishan
2 // @date : 20/08/2023
3 // @time : 10:00 AM
4 // @category : Public
5 // @version : 1.0
6 // @description : This is a public category.
7 // @license : MIT
8 // @url : https://github.com/KunalKishan/My-Portfolio
9 // @email : kunal.kishan@gmail.com
10 // @phone : +91 98989 98989
11 // @address : Kunal Kishan, 123, Main Street, New York, NY 10001, USA
12 // @website : https://kunal.kishan.com
13 // @social : https://www.linkedin.com/in/kunal-kishan/
14 // @github : https://github.com/KunalKishan
15 // @twitter : https://twitter.com/KunalKishan
16 // @facebook : https://facebook.com/KunalKishan
17 // @instagram : https://www.instagram.com/kunal.kishan/
18 // @youtube : https://www.youtube.com/channel/UCkunal_kishan
19 // @medium : https://medium.com/@kunal.kishan
20 // @dev.to : https://dev.to/kunal.kishan
21 // @leetcode : https://leetcode.com/kunal.kishan
22 // @codecademy : https://www.codecademy.com/profile/kunal.kishan
23 // @freeCodeCamp : https://www.freecodecamp.org/profile/kunal.kishan
24 // @kaggle : https://www.kaggle.com/kunal.kishan
25 // @datacamp : https://www.datacamp.com/profile/kunal.kishan
26 // @coursera : https://www.coursera.org/profile/kunal.kishan
27 // @edx : https://www.edx.org/profile/kunal.kishan
28 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
29 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
30 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
31 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
32 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
33 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
34 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
35 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
36 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
37 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
38 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
39 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
40 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
41 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
42 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
43 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
44 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
45 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
46 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
47 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
48 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
49 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
50 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
51 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
52 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
53 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
54 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
55 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
56 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
57 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
58 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
59 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
60 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
61 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
62 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
63 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
64 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
65 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
66 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
67 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
68 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
69 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
70 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
71 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
72 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
73 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
74 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
75 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
76 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
77 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
78 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
79 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
80 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
81 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
82 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
83 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
84 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
85 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
86 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
87 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
88 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
89 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
90 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
91 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
92 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
93 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
94 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
95 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
96 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
97 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan
98 // @blackboard : https://www.blackboard.com/profile/kunal.kishan
99 // @canvas : https://www.canvaslms.com/profile/kunal.kishan
100 // @futurelearn : https://www.futurelearn.com/profile/kunal.kishan

```

ID	Kode Kategori	Nama Kategori
1	ATM	Alat Makan
2	PRK	Perkakas
3	MNM	Mimman
4	PKB	Perlengkapan Beagke
5	TKN	Teleponologi

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	 Edit  Copy  Delete	1	ATM	Alat Makan	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	2	PRK	Perkakas	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	3	MNM	Minuman	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	4	PKB	Perlengkapan Bengkel	NULL	NULL
<input type="checkbox"/>	 Edit  Copy  Delete	5	TKN	Teknologi	NULL	NULL

- Hal. 20 / 23



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

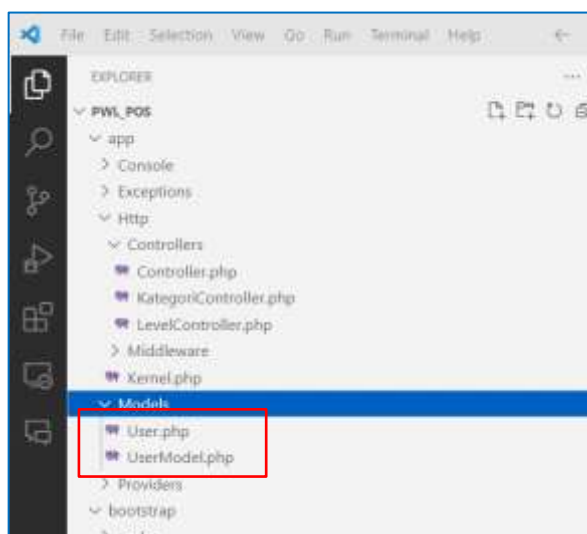
```
php artisan make:model <nama-model-CamelCase>
```

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`



```
1 <doctype html>
2 <html>
3 <head>
4 <title><data user/>title</data>
5 </head>
6 <body>
7 <div data user/>
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 <tr>
16 <td><data as $id>
17 <td>< $id-user_id ></td>
18 <td>< $id-username ></td>
19 <td>< $id-nama ></td>
20 <td>< $id-level_id ></td>
21 </tr>
22 </table>
23 </div>
24 </body>
25 </html>
```

localhost/PWL2023/pwl_week3/PWLPO5/public/user

Data User

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

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$hc54fM27UjXxmZeiOK7aJAdCYybKyNLIk8BleydN/...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$VProOGCadKgJwAOK8G0xpIFC9IAEaAyOu9h3vjRO4r...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Amdxg9eUSdyzj9BC8561sewJRI602dTtW10SIAYrp...	NULL	NULL

7. Jalankan di browser, catat dan laporkan apa yang terjadi
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
10. Kita modifikasi lagi file `UserController` menjadi seperti berikut

```
1 class UserController extends Controller
2 {
3     public function index()
4     {
5         // tambah data user dengan Eloquent Model
6         $data = [
7             'nama' => 'Pelanggan Pertama',
8         ];
9         UserModel::where('username', 'customer-1')->update($data); // update data user
10
11         // coba akses model UserModel
12         $user = UserModel::all(); // ambil semua data dari tabel m_user
13         return view('user', ['data' => $user]);
14     }
15 }
```




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

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

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/>	1	1	admin	Administrator	\$2y\$12\$e34BZ77U00nUwOK7aJAdYyMfALP8D9eNE	NULL	NULL
<input type="checkbox"/>	2	2	manager	Manager	\$2y\$12\$VPs00GdKglwAK80x0FCBAE5AyDvM0uR04	NULL	NULL
<input type="checkbox"/>	3	2	staff	StaffKasir	\$2y\$12\$Amdg6eU5dyg8D0856fueyUM6328TTt2WY0SRyp	NULL	NULL

G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari `APP_KEY` pada *file setting .env* Laravel? `APP_KEY` adalah token keamanan, yang memastikan bahwa data yang dienkripsi tidak dapat dengan mudah di decrypt tanpa kunci yang sesuai.
2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk `APP_KEY`? Menggunakan perintah `php artisan key:generate`
3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
 - `Create_user_table` : Membuat tabel user
 - `Create_password_resets_table` : menyimpan reset password
 - `Create_failed_jobs_table` : untuk mencatat yang gagal
4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/output dari fungsi tersebut?
 - `Create_personal_access_tokens_table` : membuat tabel personal access yang menyimpan token API saat menggunakan laravel authentication
5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
Membuat kolom primary key dengan id, yang dihasilkan adalah big integer dengan auto increment
6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?
`$table->id();` : membuat kolom id sebagai primary key
`$table->id('level_id');` : membuat kolom level_id sebagai primary key
7. Pada migration, Fungsi `->unique()` digunakan untuk apa? Memastikan nilai pada sebuah kolom tidak ada yang duplikat.
8. Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom `level_id` pada tabel `m_user` menggunakan `$table->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$table->id('level_id')` ?
Dikarenakan pada tabel `m_user` mereferensikan foreign key dari tabel `m_level`, sedangkan pada tabel `m_level` tidak perlu mereferensikan foreign key dari tabel lain.



9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?
Hash digunakan untuk mengenkripsi password sebelum disimpan ke database
Make('1234') mengubah 1234 menjadi hash bcrypt untuk keamanan
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
Sebagai parameter binding untuk keamanan saat memasukkan nilai, mencegah SQL injection
11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';`?
- `protected $table = 'm_user';` → Menentukan bahwa model ini menggunakan tabel `m_user`.
- `protected $primaryKey = 'user_id';` → Menentukan primary key yang digunakan adalah `user_id`, bukan `id` (default).
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Facade / Query Builder / Eloquent ORM*) ? jelaskan
DB Facade : lebih cepat namun lebih kompleks dikarenakan query harus ditulis manual
Query Builder: lebih fleksibel cocok untuk query yang kompleks tanpa harus menggunakan raw sql
Eloquent ORM : Paling mudah karena berbasis OOP, tetapi lebih lambat dibanding Query Builder untuk query sederhana

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