

Laporan Pemrograman Web Lanjut

Jobsheet 4 : MODEL dan ELOQUENT ORM

Dosen Pengampu : Dimas Wahyu Wibowo, ST.,MT.



Disusun Oleh:

Queenadhynar Azarine Dwipa A.

2341760109

SIB 2B

JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2023/2024

A. Properti \$fillable dan \$guarded

Praktikum 1 - \$fillable

1. Buka file model dengan nama UserModel.php dan tambahkan \$fillable seperti gambar di bawah ini

```
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user'; // mendefinisikan nama tabel yang akan digunakan
    protected $primaryKey = 'user_id'; // mendefinisikan primary key dari tabel

    protected $fillable = ['level_id', 'username', 'nama', 'password'];
}
```

2. Buka file controller dengan nama UserController.php dan ubah script untuk menambahkan data baru seperti gambar di bawah ini

```
public function index()
{
    // tambah data user dengan $fillable
    $data = [
        'level_id' => 2,
        'username' => 'manager_dua',
        'nama' => 'Manager 2',
        'password' => Hash::make('12345');
    ];
    UserModel::create($data);

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

3. Simpan kode program Langkah 1 dan 2, dan jalankan perintah web server. Kemudian jalankan link localhostPWL_POS/public/user pada browser dan amati apa yang terjadi

Data User

ID	Username	Nama	IDPengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
15	customer-1	Pelanggan Pertama	5
17	manager_dua	Manager 2	2

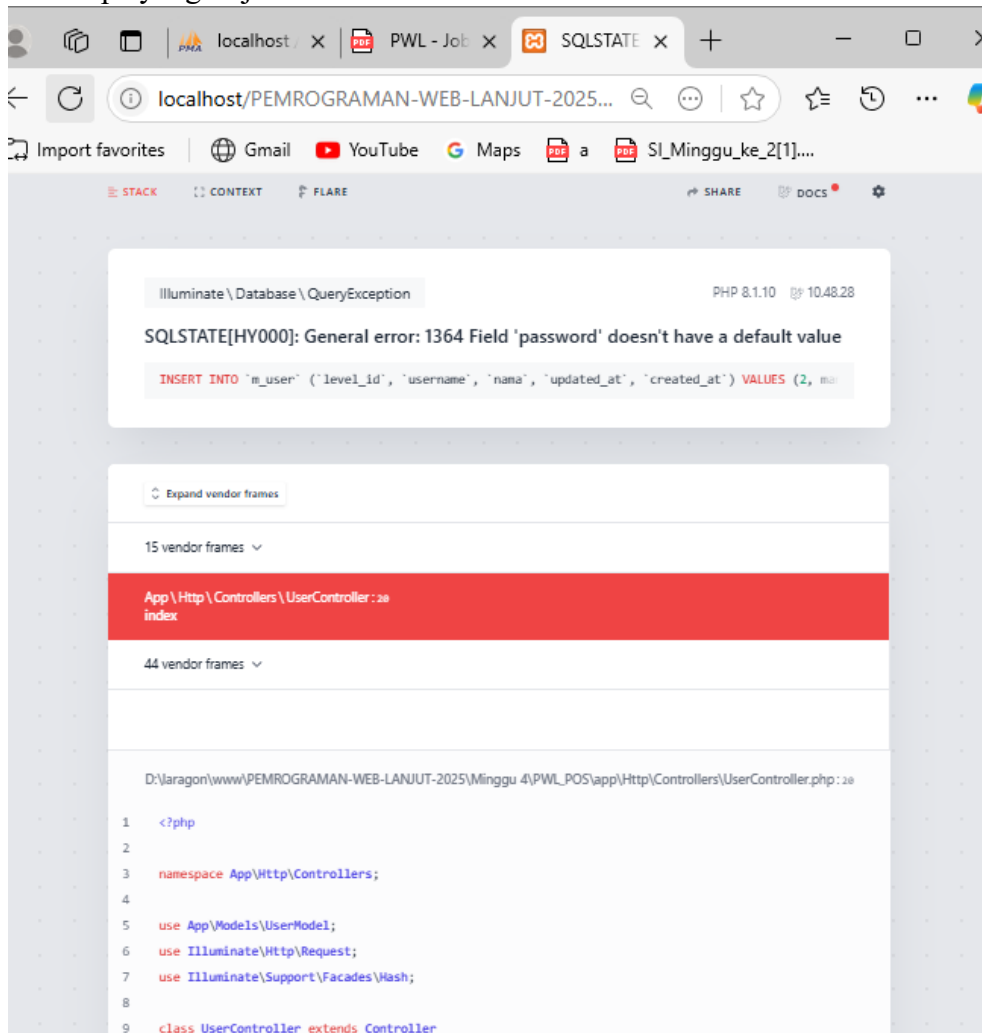
4. Ubah file model UserModel.php seperti pada gambar di bawah ini pada bagian \$fillable

```
protected $fillable = ['level_id', 'username', 'nama'];
```

5. Ubah kembali file controller UserController.php seperti pada gambar di bawah hanya bagian array pada \$data

```
// tambah data user dengan Eloquent Model
$data = [
    'level_id' => 2,
    'username' => 'manager_tiga',
    'nama' => 'Manager 3',
    'password' => Hash::make('12345')
];
UserModel::create($data);
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan pada browser dan amati apa yang terjadi



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

\$guarded

Kebalikan dari \$fillable adalah \$guarded. Semua kolom yang kita tambahkan ke \$guarded akan diabaikan oleh Eloquent ketika kita melakukan insert/update. Secara default \$guarded isinya array(""), yang berarti semua atribut tidak bisa diset melalui mass assignment. Mass Assignment adalah fitur canggih yang menyederhanakan proses pengaturan beberapa atribut model sekaligus, menghemat waktu dan tenaga. Pada praktikum ini, kita akan mengeksplorasi konsep penugasan

massal di Laravel dan bagaimana hal itu dapat dimanfaatkan secara efektif untuk meningkatkan alur kerja pengembangan Anda.

B. Retrieving Single Models

Praktikum 2.1 – Retrieving Single Models

1. Buka file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::find(1);
        return view('user', ['data' => $user]);
    }
}
```

2. Buka file view dengan nama user.blade.php dan ubah script seperti gambar di bawah ini

```
<body>
<h1>Data User</h1>
<table border="1" cellpadding="2">
<tr>
<th>ID</th>
<th>Username</th>
<th>Nama</th>
<th>IDPengguna</th>
</tr>
<tr>
<td>{{ $data->user_id }}</td>
<td>{{ $data->username }}</td>
<td>{{ $data->nama }}</td>
<td>{{ $data->level_id }}</td>
</tr>
</table>
</body>
</html>
```

3. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

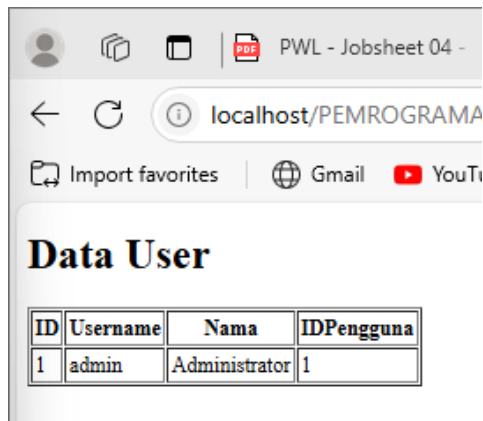
Data User

ID	Username	Nama	IDPengguna
1	admin	Administrator	1

- Kode diatas berfungsi untuk menampilkan data pengguna dalam tabel
 - UserModel::find(1) mengambil data pengguna berdasarkan id 1 dari database
4. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::where('level_id', 1)->first();
return view('user', ['data' => $user]);
```

5. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan



➤ Kode diatas mencari data berdasarkan kolom tertentu (level_id)

- Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::firstWhere('level_id', 1);
```

- Simpan kode program Langkah 6. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

ID	Username	Nama	IDPengguna
1	admin	Administrator	1

➤ Kode diatas mencari data pertama yang memiliki level_id = 1 dalam table, lebih ringkas dibanding dengan where()->first() namun fungsinya sama. Terkadang Anda mungkin ingin melakukan beberapa tindakan lain jika tidak ada hasil yang ditemukan. Metode findOr and firstOr akan mengembalikan satu contoh model atau, jika tidak ada hasil yang ditemukan maka akan menjalankan didalam fungsi. Nilai yang dikembalikan oleh fungsi akan dianggap sebagai hasil dari metode ini:

```
$user = UserModel::findOr(1, function () {
    // ...
});

$user = UserModel::where('level_id', '>', 3)->firstOr(function () {
    // ...
});
```

- Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::findOr(1, ['username', 'nama'], function () {
    abort(484);
});
return view('user', ['data' => $user]);
```

9. Simpan kode program Langkah 8. Kemudian pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan.

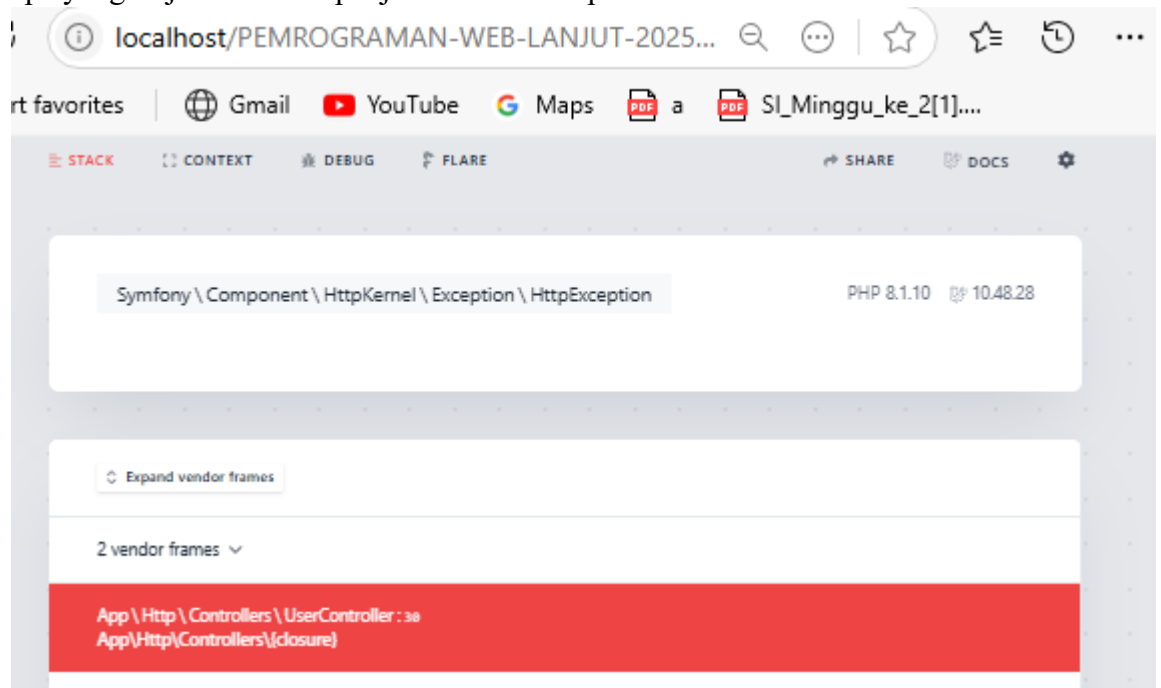
Data User

ID	Username	Nama	IDPengguna
	admin	Administrator	

- Kode digunakan untuk mencari data pengguna berdasarkan id = 1. Jika ditemukan, hanya username dan nama yang diambil
 - Jika error maka kode menghentikan proses dan menampilkan error HTTP 484
10. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::findOr(20, ['username', 'nama'], function () {  
    abort(484);  
});  
return view('user', ['data' => $user]);
```

11. Simpan kode program Langkah 10. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan



- Hasil kode diatas *error* dikarenakan tidak menemukan id = 20 didalam tabel Database
12. Laporkan hasil Praktikum-2.1 ini dan commit perubahan pada git.

Praktikum 2.2 – Not Found Exceptions

Terkadang Anda mungkin ingin memberikan pengecualian jika model tidak ditemukan. Hal ini sangat berguna dalam rute atau pengontrol. Metode `findOrFail` and `firstOrFail` akan mengambil hasil pertama dari kueri; namun, jika tidak ada hasil yang ditemukan, sebuah `Illuminate\Database\Eloquent\ModelNotFoundException` akan dilempar. Berikut ikuti langkah-langkah di bawah ini:

1. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
public function index()  
{  
    $user = UserModel::findOrFail(1);  
    return view('user', ['data' => $user]);  
}
```

2. Simpan kode program Langkah 1. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

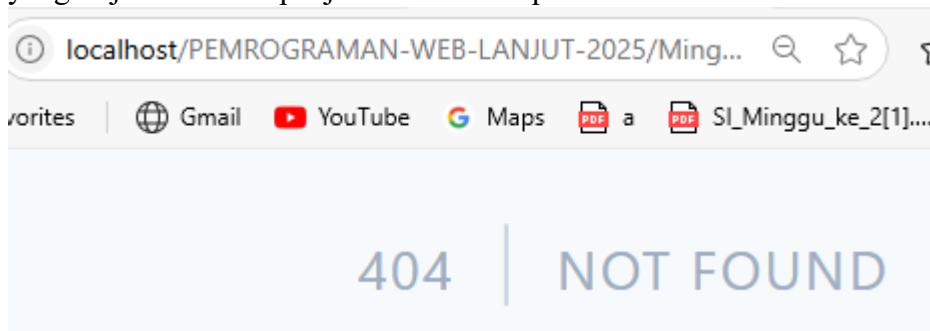
Data User

ID	Username	Nama	IDPengguna
1	admin	Administrator	1

- Kode diatas untuk mencari data berdasarkan id = 1 dalam DB
Jika ditemukan maka dikembalikan sebagai objek \$user
Jika tidak ditemukan, maka akan menampilkan error 404 (Not Found)
3. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::where('username', 'manager9')->firstOrFail();  
return view('user', ['data' => $user]);
```

4. Simpan kode program Langkah 3. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

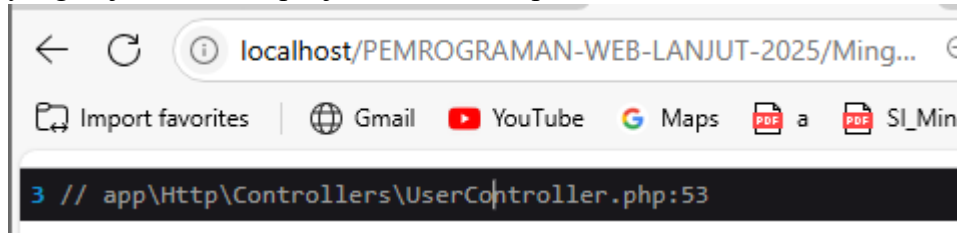


- Karena kode tidak menemukan username manager9 pada tabel maka menampilkan error 404(NOT FOUND)
5. Laporkan hasil Praktikum-2.2 ini dan commit perubahan pada git.

Praktikum 2.3 – Retrieving Aggregates

Saat berinteraksi dengan model Eloquent, Anda juga dapat menggunakan metode agregat count, sum, max, dan lainnya yang disediakan oleh pembuat kueri Laravel. Seperti yang Anda duga, metode ini mengembalikan nilai skalar dan contoh model Eloquent:

1. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini
2. Simpan kode program Langkah 1. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan



- Kode diatas menghitung jumlah pengguna dengan level_id = 2
3. Buat agar jumlah script pada langkah 1 bisa tampil pada halaman browser, sebagai contoh bisa lihat gambar di bawah ini dan ubah script pada file view supaya bisa muncul datanya

- User Controller

```
$user = UserModel::where('level_id', 2)->count();  
return view('user', ['data' => $user]); // Mengirim data ke view
```

- User Blade

```
<tr>  
    <th>Jumlah Pengguna</th> // Praktikum 2.3  
</tr>  
<tr>  
    <td>{{ $data }}</td>  
</tr>
```

4. Laporkan hasil Praktikum-2.3 ini dan commit perubahan pada git.

Data User

Jumlah Pengguna
3

Praktikum 2.4 – Retrieving or Creating Models

Metode firstOrCreate merupakan metode untuk melakukan retrieving data (mengambil data) berdasarkan nilai yang ingin dicari, jika data tidak ditemukan maka method ini akan melakukan insert ke table database tersebut sesuai dengan nilai yang dimasukkan.

Metode firstOrCreate, seperti firstOrCreate, akan mencoba menemukan/mengambil record/data dalam database yang cocok dengan atribut yang diberikan. Namun, jika data tidak ditemukan, data akan disiapkan untuk di-insert-kan ke database dan model baru akan dikembalikan. Perhatikan bahwa model yang dikembalikan firstOrCreate belum disimpan ke database. Anda perlu memanggil metode save() secara manual untuk menyimpannya:

1. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
//Praktikum 2.3 - Retrieving or Creating Models
$user = UserModel::firstOrCreate(
    [
        'username' => 'manager',
        'nama' => 'Manager',
    ],
);
return view('user', ['data' => $user]);
```

2. Ubah kembali file view dengan nama user.blade.php dan ubah script seperti gambar di bawah ini

```
<tr>
    <th>ID</th>
    <th>Username</th>
    <th>Nama</th>
    <th>IDPengguna</th>
</tr>
<tr>
    <td>{{ $data->user_id }}</td>
    <td>{{ $data->username }}</td>
    <td>{{ $data->nama }}</td>
    <td>{{ $data->level_id }}</td>
</tr>
```

3. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

ID	Username	Nama	IDPengguna
2	manager	Manager	2

➤ Fungsi firstOrCreate

Fungsi ini akan mencari data pertama sesuai dengan kriteria yang diberikan, jika data tidak ada maka akan dibuat entri baru dengan nilai yang diberikan.

4. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```

$user = UserModel::firstOrCreate(
    [
        'username' => 'manager22',
        'nama' => 'Manager Dua Dua',
        'password' => Hash::make('12345'),
        'level_id' => 2
    ],
);
return view('user', ['data' => $user]);

```

5. Simpan kode program Langkah 4. Kemudian jalankan pada browser dan amati apa yang terjadi dan cek juga pada phpMyAdmin pada tabel m_user serta beri penjelasan dalam laporan
 - Pada browser saat dijalankan

Data User

ID	Username	Nama	IDPengguna
19	manager22	Manager Dua Dua	2

- Pada table m_user

<input type="checkbox"/>	Edit	Copy	Delete	2	2 manager	Manager	\$2y\$12\$L0n3SRRVxqvrL8HkLi.4teLFAOvF1SifgzdC4GA
<input type="checkbox"/>	Edit	Copy	Delete	3	3 staff	Staff/Kasir	\$2y\$12\$v6365NjKLxaAjZrBLUBqH.yShSjbl81R.7rZ4vR5A
<input type="checkbox"/>	Edit	Copy	Delete	15	5 customer-1	Pelanggan Pertama	\$2y\$12\$VZBM8CerFzFW1eMgRtH5CauVdJGZmrYpw09TF
<input type="checkbox"/>	Edit	Copy	Delete	17	2 manager_dua	Manager 2	\$2y\$12\$udbzy9sM8E2NICIYQTSIOu35MybgaKmkIq.bpD8
<input type="checkbox"/>	Edit	Copy	Delete	18	2 manager_tiga	Manager 3	\$2y\$12\$Yf.weiRnsYKqnkZRd0BjueNIJXZq1iktDqwTKUKC
<input type="checkbox"/>	Edit	Copy	Delete	19	2 manager22	Manager Dua Dua	\$2y\$12\$ZjoLDqMnxTobQtaVIR53Eu0tW0wCzTvUye3SIXj

- Seperti langkah sebelumnya fungsi tersebut akan mencari database berdasarkan kondisi yang diberikan. Karena data tersebut belum ada maka akan dibuat entri baru dalam database menggunakan nilai yang diberikan. Seperti contohnya data diatas menambahkan **username manager** dengan **Nama Manager Dua Dua** dan **level id 2** kedalam tabel **m_user**
6. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```

$user = UserModel::firstOrCreate(
    [
        'username' => 'manager',
        'nama' => 'Manager',
    ],
);
return view('user', ['data' => $user]);

```

7. Simpan kode program Langkah 6. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

ID	Username	Nama	IDPengguna
2	manager	Manager	2



























- Fungsi firstOrCreate akan memberikan data dalam database berdasarkan kondisi yang diberikan. Jika data ditemukan maka akan mengembalikan objek model yang sesuai, jika tidak maka akan membuat objek model baru di memori tidak dalam database
8. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::firstOrCreate(
[
    'username' => 'manager33',
    'nama' => 'Manager Tiga Tiga',
    'password' => Hash::make('12345'),
    'level_id' => 2
],
);
return view('user', ['data' => $user]);
```

9. Simpan kode program Langkah 8. Kemudian jalankan pada browser dan amati apa yang terjadi dan cek juga pada phpMyAdmin pada tabel m_user serta beri penjelasan dalam laporan

Data User

ID	Username	Nama	IDPengguna
	manager33	Manager Tiga Tiga	2

				user_id	level_id	username	nama	password
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	1	admin	Administrator	\$2y\$12\$GbatlXYqAMInGJAHV
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2	manager	Manager	\$2y\$12\$L0n3SRRVxqvrL8HkLi
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	3	staff	Staff/Kasir	\$2y\$12\$5v5365NjKLxaAjZrBLUI
<input type="checkbox"/>	 Edit	 Copy	 Delete	15	5	customer-1	Pelanggan Pertama	\$2y\$12\$VZBM8CerFzFW1eMg
<input type="checkbox"/>	 Edit	 Copy	 Delete	17	2	manager_dua	Manager 2	\$2y\$12\$Sudby9sM8E2NICIQY
<input type="checkbox"/>	 Edit	 Copy	 Delete	18	2	manager_tiga	Manager 3	\$2y\$12\$Yf.weiRnsYKqnkZRd0
<input type="checkbox"/>	 Edit	 Copy	 Delete	19	2	manager22	Manager Dua Dua	\$2y\$12\$ZjoLDqMnxTcbQtaVIR
	<input type="checkbox"/> Check all	With selected:	 Edit	 Copy	 Delete	 Export		

- Seperti langkah sebelumnya fungsi diatas mengembalikan objek model yang sesuai. Data diatas tidak ditemukan namun tidak langsung menyimpan didalam database namun disimpan di memori
10. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```

$user = UserModel::firstOrCreate(
    [
        'username' => 'manager33',
        'nama' => 'Manager Tiga Tiga',
        'password' => Hash::make('12345'),
        'level_id' => 2
    ],
);
$user->save();

```

11. Simpan kode program Langkah 9. Kemudian jalankan pada browser dan amati apa yang terjadi dan cek juga pada phpMyAdmin pada tabel m_user serta beri penjelasan dalam laporan

Data User

ID	Username	Nama	IDPengguna
20	manager33	Manager Tiga Tiga	2

<input type="checkbox"/>	 Edit	 Copy	 Delete	18	2	manager_tiga	Manager 3	\$2y\$12\$Yf.weiRnsYKqnkZRd0BjueNiJXZq1ikt
<input type="checkbox"/>	 Edit	 Copy	 Delete	19	2	manager22	Manager Dua Dua	\$2y\$12\$ZjoLDqMnxTcbQtaVIR53Eu0tW0wCz
<input type="checkbox"/>	 Edit	 Copy	 Delete	20	2	manager33	Manager Tiga Tiga	\$2y\$12\$Y7B2BSukZavR1xPLjQgYw.qTxECSt

- Setelah menambahkan save() maka data yang ditambahkan akan langsung menyimpan ke dalam database. Seperti hasil diatas **manager33** yang sebelumnya tidak tersimpan dalam database akan langsung muncul dan tersimpan

12. Laporkan hasil Praktikum-2.4 ini dan commit perubahan pada git.

Praktikum 2.5 – Attribute Changes

Eloquent menyediakan metode `isDirty`, `isClean`, dan `wasChanged` untuk memeriksa keadaan internal model Anda dan menentukan bagaimana atributnya berubah sejak model pertama kali diambil.

Metode `isDirty` menentukan apakah ada atribut model yang telah diubah sejak model diambil. Anda dapat meneruskan nama atribut tertentu atau serangkaian atribut ke metode `isDirty` untuk menentukan apakah ada atribut yang "kotor". Metode `isClean` akan menentukan apakah suatu atribut tetap tidak berubah sejak model diambil. Metode ini juga menerima argumen atribut opsional:

➤ Kode

```
$user = UserModel::create([
    'username' => 'manager44',
    'nama' => 'Manager44',
    'password' => Hash::make('12345'),
    'level_id' => 2
]);
$user->username = 'manager45';

$user->isDirty(); //true
$user->isDirty('username'); //true
$user->isDirty('false'); //false
$user->isDirty(['nama', 'username']); //true

$user->isClean(); //false
$user->isClean('username'); //false
$user->isClean('nama'); //false
$user->isClean(['nama', 'username']); //false

$user->save();

$user->isDirty(); //false
$user->isClean(); //true
```

➤ Hasil dari Kode diatas

<input type="checkbox"/>	 Edit	 Copy	 Delete	19	2	manager22	Manager Dua Dua	\$2y\$12\$ZjLDqMnxTobQtaVIR53Eu0tW0wCzTvUye3SiXy
<input type="checkbox"/>	 Edit	 Copy	 Delete	20	2	manager33	Manager Tiga Tiga	\$2y\$12\$Y7B2BSukZavR1xPLjQgYw.qTxECs9fm5o4NHlt
<input type="checkbox"/>	 Edit	 Copy	 Delete	21	2	manager45	Manager44	\$2y\$12\$Sr46lhpXuAdZaZ5ueWzgxHO/ftOqeHI7u4BHPf5ld
<input type="checkbox"/>	 Edit	 Copy	 Delete	22	2	manager44	Manager44	\$2y\$12\$J0jQOIptmgS7KC0.PK8Si.z7OcuJ4MMKqGq.BImV

1. Ubah file controller dengan nama `UserController.php` dan ubah script seperti gambar di bawah ini

```
//Praktikum 2.5 – Attribute Changes
$user = UserModel::create([
    'username' => 'manager55',
    'nama' => 'Manager55',
    'password' => Hash::make('12345'),
    'level_id' => 2
]);
$user->username = 'manager56';

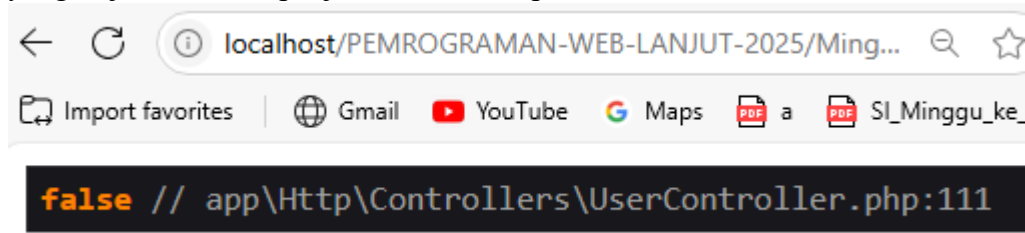
$user->isDirty(); //true
$user->isDirty('username'); //true
$user->isDirty('false'); //false
$user->isDirty(['nama', 'username']); //true

$user->isClean(); //false
$user->isClean('username'); //false
$user->isClean('nama'); //true
$user->isClean(['nama', 'username']); //false

$user->save();

$user->isDirty(); //false
$user->isClean(); //true
dd($user->isDirty());
```

2. Simpan kode program Langkah 1. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan



- False muncul karena tidak ada perubahan yang belum disimpan, jika isDirty() dipanggil setelah save maka hasilnya pasti false.
- Jika nilai baru sama dengan yang lama, Laravel tidak menganggap sebagai perubahan

Metode ini wasChanged menentukan apakah ada atribut yang diubah saat model terakhir disimpan dalam siklus permintaan saat ini. Jika diperlukan, Anda dapat memberikan nama atribut untuk melihat apakah atribut tertentu telah diubah:

```
$user = UserModel::create([
    'username' => 'manager11',
    'nama' => 'Manager11',
    'password' => Hash::make('12345'),
    'level_id' => 2
]);
$user->username = 'manager12';
$user->save();

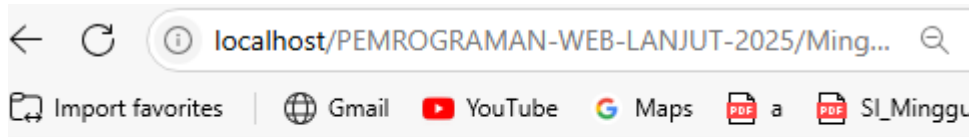
$user->wasChanged(); //true
$user->wasChanged('username'); //true
$user->wasChanged(['username', 'level_id']); //true
$user->wasChanged('nama'); //false
$user->wasChanged(['nama', 'username']); //true
```

3. Ubah file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

```
$user = UserModel::create([
    'username' => 'manager11',
    'nama' => 'Manager11',
    'password' => Hash::make('12345'),
    'level_id' => 2
]);
$user->username = 'manager12';
$user->save();

$user->wasChanged(); //true
$user->wasChanged('username'); //true
$user->wasChanged(['username', 'level_id']); //true
$user->wasChanged('nama'); //false
dd($user->wasChanged(['nama', 'username'])); //true
```

4. Simpan kode program Langkah 3. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan



```
true // app\Http\Controllers\UserController.php:132
```

- wasChanged() mengecek apakah atribut berubah setelah pemanggilan save(). isDirty() digunakan sebelum save() sedangkan wasChanged digunakan setelah save(). Hasil true muncul karena username berubah dari 'manager11' -> 'manager'. Jika atribut yang diperiksa tidak berubah (nama) hasilnya akan fake

5. Laporkan hasil Praktikum-2.5 ini dan commit perubahan pada git.

Praktikum 2.6 – Create, Read, Update, Delete (CRUD)

Seperti yang telah kita ketahui, CRUD merupakan singkatan dari Create, Read, Update dan Delete. CRUD merupakan istilah untuk proses pengolahan data pada database, seperti input data ke database, menampilkan data dari database, mengedit data pada database dan menghapus data dari database. Ikuti langkah-langkah di bawah ini untuk melakukan CRUD dengan Eloquent

1. Buka file view pada user.blade.php dan buat scriptnya menjadi seperti di bawah ini

```
<body>
<h1>Data User</h1>
<a href="/user/tambah">+ Tambah User</a>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
    <td>ID</td>
    <td>Username</td>
    <td>Nama</td>
    <td>IDPengguna</td>
  </tr>
  @foreach ($data as $d)
  <tr>
    <td>{{ $d->user_id }}</td>
    <td>{{ $d->username }}</td>
    <td>{{ $d->nama }}</td>
    <td>{{ $d->level_id }}</td>
    <td>
      <a href="/user/ubah/{{ $d->user_id }}">Ubah</a> |
      <a href="/user/hapus/{{ $d->user_id }}">Hapus</a>
    </td>
  </tr>
  @endforeach
</table>
</body>
```

2. Buka file controller pada UserController.php dan buat scriptnya untuk read menjadi seperti di bawah ini

```
//Praktikum 2.6 - Create, Read, Delete (CRUD)
$user = UserModel::all();
return view('user', ['data' => $user]);
```

3. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

[+ Tambah User](#)

ID	Username	Nama	IDPengguna	
1	admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff/Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus

- Kode diatas menampilkan daftar pengguna dalam tabel dan menyediakan fitur ubah serta hapus
4. Langkah berikutnya membuat create atau tambah data user dengan cara bikin file baru pada view dengan nama user_tambah.blade.php dan buat scriptnya menjadi seperti di bawah ini

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Form</title>
</head>
<body>
  <h1>Form Tambah Data User</h1>
  <form method="post" action="/user/tambah_simpan">

    {{ csrf_field() }}

    <label>Username</label>
    <input type="text" name='username' placeholder="Masukkan Username">
    <br>
    <label>Nama</label>
    <input type="text" name='nama' placeholder="Masukkan Nama">
    <br>
    <label>Password</label>
    <input type="password" name='password' placeholder="Masukkan Password">
    <br>
    <label>Levek ID</label>
    <input type="number" name='level_id' placeholder="Masukkan ID Level">
    <br><br>
    <input type="submit" class="btn btn-success" value="Simpan">

  </form>
</body>
</html>
```


5. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
Route::get('/user/tambah', [UserController::class, 'tambah']);
```

6. Tambahkan script pada controller dengan nama file UserController.php. Tambahkan script dalam class dan buat method baru dengan nama tambah dan diletakan di bawah method index seperti gambar di bawah ini

```
//Praktikum 2.6 - Create, Read, Delete (CRUD)
$user = UserModel::all();
return view('user', ['data' => $user]);

public function tambah()
{
    return view("user_tambah");
}
```

7. Simpan kode program Langkah 4 s/d 6. Kemudian jalankan pada browser dan klik link “+ Tambah User” amati apa yang terjadi dan beri penjelasan dalam laporan

Form Tambah Data User

Username

Nama

Password

Level ID

- Kode ini menyediakan form untuk menambahkan user dengan atribut username, nama, password dan level ID. Lalu route GET /user/tambah digunakan untuk menampilkan halaman form. Metode tambah() hanya merender tampilan form user_tambah.blade.php

8. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
```

9. Tambahkan script pada controller dengan nama file UserController.php. Tambahkan script dalam class dan buat method baru dengan nama tambah_simpan dan diletakan di bawah method tambah seperti gambar di bawah ini

```
public function tambah_simpan(Request $request)
{
    UserModel::create([
        'username' => $request->username,
        'nama' => $request->nama,
        'password' => Hash::make($request->password),
        'level_id' => $request->level_id
    ]);

    return redirect('/user');
}
```

10. Simpan kode program Langkah 8 dan 9. Kemudian jalankan link localhost:8000/user/tambah atau localhost/PWL_POS/public/user/tambah pada browser dan input formnya dan simpan, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

Form Tambah Data User

Username

Nama

Password

Level ID

Data User

[+ Tambah User](#)

ID	Username	Nama	IDPengguna	
1	admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff/Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
25	manager66	Manager66	2	Ubah Hapus

- Metode Request menerima objek yang berisi data dari formular yang dikirim oleh pengguna. Data berasal dari input pada form HTML yang sebelumnya telah dibuat
 - UserModel::create() digunakan untuk menyimpan data ke dalam database
11. Langkah berikutnya membuat update atau ubah data user dengan cara bikin file baru pada view dengan nama user_ubah.blade.php dan buat scriptnya menjadi seperti di bawah ini

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Form</title>
</head>
<body>
<h1>Form Ubah Data User</h1>
<a href="/user">Kembali</a>
<form method="post" action="/PEMROGRAMAN-WEB-LANJUT-2025/MingguX284/PWL_POS/public/user/ubah_simpan/{{ $data->user_id }}">
    {{ csrf_field() }}
    {{ method_field('PUT') }}

    <label>Username</label>
    <input type="text" name="username" placeholder="Masukkan Username" value="{{ $data->username }}">
    <br>
    <label>Nama</label>
    <input type="text" name="nama" placeholder="Masukkan Nama" value="{{ $data->nama }}">
    <br>
    <label>Password</label>
    <input type="password" name="password" placeholder="Masukkan Password" value="{{ $data->password }}">
    <br>
    <label>Level ID</label>
    <input type="number" name="level_id" placeholder="Masukkan ID Level" value="{{ $data->level_id }}">
    <br><br>
    <input type="submit" class="btn btn-success" value="Ubah">
</form>
</body>
</html>
```

12. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
Route::get('/user/ubah/{id}', [UserController::class, 'ubah']);
```

13. Tambahkan script pada controller dengan nama file UserController.php. Tambahkan script dalam class dan buat method baru dengan nama ubah dan diletakan di bawah method tambah_simpan seperti gambar di bawah ini 14. Simpan kode program Langkah 11 sd

```
public function ubah($id)
{
    $user = UserModel::find($id);
    return view('user_ubah', ['data' => $user]);
}
```

14. Kemudian jalankan pada browser dan klik link “Ubah” amati apa yang terjadi dan beri penjelasan dalam laporan
- Saat di klik link “Ubah” maka akan masuk ke /ubah/{id}

Form Ubah Data User

[Kembali](#)

Username

Nama

Password

Level ID

- Saat di klik Kembali, maka akan Kembali ke page /user

Data User

[+ Tambah User](#)

ID	Username	Nama	IDPengguna	
1	admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff/Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
25	manager66	Manager66	2	Ubah Hapus

- Diatas menampilkan form edit user dengan data yang sudah ada, menggunakan metode PUT untuk memperbarui data di database. Route Get digunakan untuk menampilkan halaman edit berdasarkan ID pengguna

15. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);
```

16. Tambahkan script pada controller dengan nama file UserController.php. Tambahkan script dalam class dan buat method baru dengan nama ubah_simpan dan diletakan di bawah method ubah seperti gambar di bawah ini

```
public function ubah_simpan($id, Request $request)
{
    $user = UserModel::find($id);

    $user->username = $request->username;
    $user->nama = $request->nama;
    $user->password = Hash::make($request->password);
    $user->level_id = $request->level_id;

    $user->save();
    return redirect('/user');
}
```

17. Simpan kode program Langkah 15 dan 16. Kemudian jalankan link localhost:8000/user/ubah/1 atau localhost/PWL_POS/public/user/ubah/1 pada browser dan ubah input formnya dan klik tombol ubah, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

Form Ubah Data User Form Ubah Data User

[Kembali](#)

Username

Nama

Password

Level ID

[Kembali](#)

Username

Nama

Password

Level ID

- Setelah di ubah maka tabel akan merubah sesuai dengan data yang diisikan

Data User

[+ Tambah User](#)

ID	Username	Nama	IDPengguna	
1	admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
25	manager66	Manager66	2	Ubah Hapus

Data User

[+ Tambah User](#)

ID	Username	Nama	IDPengguna	
1	Admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
25	manager66	Manager66	2	Ubah Hapus

- Metode ubah_simpan() untuk menyimpan perubahan data user ke database, Route PUT untuk menangani pembaruan data user

- Mengupdate data user berdasarkan ID yang berikan, menggunakan metode PUT sesuai standar HTTP untuk update
18. Berikut untuk langkah delete . Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
Route::get('/user/hapus/{id}', [UserController::class, 'hapus']);
```

19. Tambahkan script pada controller dengan nama file UserController.php. Tambahkan script dalam class dan buat method baru dengan nama hapus dan diletakan di bawah method ubah_simpan seperti gambar di bawah ini

```
public function hapus($id)
{
    $user = UserModel::find($id);
    $user->delete();

    return redirect('/user');
}
```

20. Simpan kode program Langkah 18 dan 19. Kemudian jalankan pada browser dan klik tombol hapus, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan 21. Laporkan hasil Praktikum-2.6 ini dan commit perubahan pada git.

Data User

+ Tambah User

ID	Username	Nama	IDPengguna	
1	Admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
25	manager66	Manager66	2	Ubah Hapus

Data User

+ Tambah User

ID	Username	Nama	IDPengguna	
1	Admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staff Kasir	3	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	Ubah Hapus
17	manager_dua	Manager 2	2	Ubah Hapus
18	manager_tiga	Manager 3	2	Ubah Hapus
19	manager22	Manager Dua Dua	2	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	Ubah Hapus
21	manager45	Manager44	2	Ubah Hapus
22	manager44	Manager44	2	Ubah Hapus
23	manager56	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus

- Metode hapus() dalam usercontroller untuk menghapus data pengguna dari database
- Menghapus user berdasarkan id dari database, mengalihkan Kembali ke daftar user setelah penghapusan

Praktikum 2.7 – Relationships

One to One

Hubungan satu-ke-satu adalah tipe hubungan database yang sangat mendasar. Misalnya, suatu Usermodel mungkin dikaitkan dengan satu model Levelmodel. Untuk mendefinisikan hubungan ini, kita akan menempatkan Levelmodel metode pada model Usermodel. Metode tersebut Levelmodel harus memanggil hasOne metode tersebut dan mengembalikan hasilnya. Metode ini hasOne tersedia untuk model Illuminate\Database\Eloquent\Model: Anda melalui kelas dasar model

Mendefinisikan Kebalikan dari Hubungan One-to-one

Jadi, kita dapat mengakses model Levelmodel dari model Usermodel kita. Selanjutnya, mari kita tentukan hubungan pada model Levelmodel yang memungkinkan kita mengakses user. Kita dapat mendefinisikan kebalikan dari suatu hasOne hubungan menggunakan belongsTo metode:

One to Many

Hubungan satu-ke-banyak digunakan untuk mendefinisikan hubungan di mana satu model adalah induk dari satu atau lebih model turunan. Misalnya, 1 kategori mungkin memiliki jumlah barang yang tidak terbatas. Seperti semua hubungan Eloquent lainnya, hubungan satu-ke banyak ditentukan dengan mendefinisikan metode pada model Eloquent Anda:

One to Many (Inverse) / Belongs To

Sekarang kita dapat mengakses semua barang, mari kita tentukan hubungan agar barang dapat mengakses kategori induknya. Untuk menentukan invers suatu hasMany hubungan, tentukan metode hubungan pada model anak yang memanggil belongsTo tersebut:

1. Buka file model pada UserModel.php dan tambahkan scripnya menjadi seperti di bawah ini

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user'; // mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primaryKey = 'user_id'; // mendefinisikan primary key dari tabel yang digunakan

    protected $fillable = ['level_id', 'username', 'nama', 'password'];

    public function level(): BelongsTo
    {
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
    }
}
```

2. Buka file controller pada UserController.php dan ubah method script menjadi seperti di bawah ini

```
$user = UserModel::with('level')->get();
dd($user);
```

3. Simpan kode program Langkah 2. Kemudian jalankan link pada browser, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

```
Illuminate\Database\Eloquent\Collection {#323 ▼ // app\Http\Controllers\UserController.php:178
  #items: array:12 [▶]
  #escapeWhenCastingToString: false
}
```

- Untuk mendefinisikan relasi ke LevelModel melalui level_id
 - Menggunakan UserModel::with('level')->get(); untuk mengambil data user beserta levelnya secara efisien
4. Buka file controller pada UserController.php dan ubah method script menjadi seperti di bawah ini

```
$user = UserModel::with('level')->get();
return view('user', ['data' => $user]);
```

5. Buka file view pada user.blade.php dan ubah script menjadi seperti di bawah ini

```
<h1>Data User</h1>
<a href="/PEMROGRAMAN-WEB-LANJUT-2025/Minggu%204/PWL_POS/public/user/tambah">+ Tambah User</a>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
    <td>ID</td>
    <td>Username</td>
    <td>Nama</td>
    <td>IDPengguna</td>
    <td>Kode Level</td>
    <td>Nama Level</td>
  </tr>
  @foreach ($data as $d)
    <tr>
      <td>{{ $d->user_id }}</td>
      <td>{{ $d->username }}</td>
      <td>{{ $d->nama }}</td>
      <td>{{ $d->level_id }}</td>
      <td>{{ $d->level->level_kode }}</td>
      <td>{{ $d->level->level_nama }}</td>
    </tr>
  </td>
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan link pada browser, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

+ [Tambah User](#)

ID	Username	Nama	IDPengguna	Kode Level	Nama Level	
1	Admin	Administrator	1	ADM	Administrator	Ubah Hapus
2	manager	Manager	2	MNG	Manager	Ubah Hapus
3	staff	Staff/Kasir	3	STF	Staff/Kasir	Ubah Hapus
15	customer-1	Pelanggan Pertama	5	CUS	Pelanggan	Ubah Hapus
17	manager_dua	Manager 2	2	MNG	Manager	Ubah Hapus
18	manager_tiga	Manager 3	2	MNG	Manager	Ubah Hapus
19	manager22	Manager Dua Dua	2	MNG	Manager	Ubah Hapus
20	manager33	Manager Tiga Tiga	2	MNG	Manager	Ubah Hapus
21	manager45	Manager44	2	MNG	Manager	Ubah Hapus
22	manager44	Manager44	2	MNG	Manager	Ubah Hapus
23	manager56	Manager55	2	MNG	Manager	Ubah Hapus
24	manager12	Manager11	2	MNG	Manager	Ubah Hapus

- Hasilnya menjadi menambahkan informasi levelnya yang diambil dari tabel level
 - \$d->level->level_kode dan \$d->level->level_nama digunakan untuk mengakses data dari tabel level yang terkait dengan user.
 - (with('level')) agar mengambil data user dan level dalam satu query, menghindari N+1 query problem.
7. Laporkan hasil Praktikum-2.7 ini dan commit perubahan pada git.