

LAPORAN PRAKTIKUM PENGEMBANGAN WEBSITE
MIGRATION, SEEDER, & FACTORY - WEEK 12



Disusun oleh:

Nama : **Elvizto** Juan Khresnanda

Nim : L0122054

Kelas : B

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA
UNIVERSITAS SEBELAS MARET

2024

I. Source Code Program

A. Materi Wajib “Migrations, Seeder, and Factory” Laravel

Di penugasan sebelumnya, dalam program pengembangan website yang menggunakan framework ‘Laravel’ dengan memanfaatkan MVC dan Routes yang saya buat sedemikian rupa yang bertemakan tentang laman atau situs yang menyediakan pelayanan “Box Movies” untuk menampilkan informasi mengenai film-film guna menambahkan pengetahuan pengguna dalam menelusuri atau sekedar mencari suatu film yang saya sediakan.

Lalu, pada penugasan minggu ke-12 ini, saya mengembangkan kembali aplikasi website berbasis Laravel yang telah saya buat tersebut. Dimana, saya mencoba untuk mengimplementasikan “**Seeder**” dan “**Factory**” untuk mengelola (mengisikan) data yang akan disimpan pada masing-masing table database dan ditampilkan di dalam website (view table). Berikut untuk penjelasan lebih jauh mengenai pengembangan program website berbasis Laravel yang saya kembangkan pada minggu kali ini:

❖ Database Migrations

🚦 Movies Table

```
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create('movies', function (Blueprint $table) {
15               $table->id();
16               $table->string('title');
17               $table->string('image')->nullable();
18               $table->decimal('rating', 3, 1)->nullable();
19               $table->string('production');
20               $table->string('director');
21               $table->date('release_date');
22               $table->string('age_restriction');
23               $table->integer('playtime');
24               $table->text('description');
25               $table->enum('status', ['available', 'unavailable', 'coming_soon']);
26               $table->timestamps();
27           });
28       }
29
30       /**
31        * Reverse the migrations.
32        */
33       public function down(): void
34       {
35           Schema::dropIfExists('movies');
36       }
37  };
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian database migration untuk mengonstruksi table baru bernama ‘**movies**’ yang

digunakan untuk mengalokasikan dan mengelola data dari masukkan pengguna ataupun pengembang yang hendak membuat atau mengubah suatu data film. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

- Bagian ini digunakan untuk membuat tabel **'movies'** dengan berbagai kolom yang diperlukan untuk menyimpan informasi tentang film. Dimana, kolom **'id'** digunakan sebagai primary key. Kolom-kolom data ini mencakup judul, gambar (opsional dengan nilai **'nullable'**), rating (opsional dengan nilai **'nullable'**), produksi, sutradara, tanggal rilis, batasan usia, durasi putar, deskripsi, dan status yang menggunakan tipe **enum** untuk mengindikasikan status ketersediaan dari film. Tabel ini juga mencakup timestamp **'created_at'** dan **'updated_at'**.

Genres Table

```
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('genres', function (Blueprint $table) {
15             $table->id();
16             $table->string('name');
17             $table->timestamps();
18         });
19     }
20
21     /**
22      * Reverse the migrations.
23      */
24     public function down(): void
25     {
26         Schema::dropIfExists('genres');
27     }
28 };
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian database migration untuk mengonstruksi table baru bernama **'genres'** yang digunakan untuk mengalokasikan dan mengelola data dari masukkan pengembang mengenai “genre” yang nantinya akan terhubung ke data **'movies'** untuk merepresentasikan jenis atau kategori film tersebut. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

- Bagian ini digunakan untuk membuat tabel **'genres'** untuk menyimpan informasi genre film. Dimana, kolom **'id'** digunakan sebagai primary key. Kolom data ini mencakup **'name'** yang menyimpan nama genrenya. Tabel ini juga mencakup timestamp **'created_at'** dan **'updated_at'**.

Pivot Movie & Genre Table

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('movie_genre', function (Blueprint $table) {
15             $table->id();
16             $table->foreignId('movie_id')->constrained()->onDelete('cascade');
17             $table->foreignId('genre_id')->constrained()->onDelete('cascade');
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists('movie_genre');
28     }
29 };
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian database migration untuk mengonstruksi table baru bernama **'movie_genre'** yang digunakan untuk menghubungkan data **'genres'** ke dalam data **'movies'** begitupula sebaliknya yang nantinya akan digunakan untuk menampilkan kesinambungan atau hubungan antar table tersebut pada website. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

- Bagian ini digunakan untuk membuat tabel pivot **'movie_genre'** untuk menghubungkan tabel **'movies'** dan **'genres'** dengan relasi many-to-many. Dimana, kolom **'id'** digunakan sebagai primary key. Kolom **'movie_id'** dan **'genre_id'** merupakan *foreign keys* yang menghubungkan ke tabel **'movies'** dan **'genres'**.
- Pada migrations ini, tabel menggunakan **'onDelete('cascade')'** untuk memastikan bahwa jika data di tabel **'movies'** atau **'genres'** dihapus, maka data terkait di tabel **'movie_genre'** juga akan dihapus. Tabel ini juga mencakup timestamp **'created_at'** dan **'updated_at'**.

Dengan demikian, ketiga migrasi database ini bekerja bersama-sama untuk membentuk struktur dasar dari database untuk sebuah aplikasi pengelolaan film. Tabel **'movies'** menyimpan informasi dasar tentang setiap film, tabel **'genres'** menyimpan informasi tentang berbagai genre, dan tabel **'movie_genre'** menghubungkan film dengan genre-nya. Struktur ini memungkinkan fleksibilitas dalam menyimpan dan mengelola informasi film, termasuk berbagai genre yang dapat dikaitkan dengan setiap film. Hal ini mempermudah pengembangan fitur-fitur aplikasi seperti filter berdasarkan genre, pengelolaan koleksi film, dan lain sebagainya.

❖ Models For 'Genres' & 'Movies'

🚦 Models 'Movie'

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Movie extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = [
13         'title', 'image', 'rating', 'production', 'director', 'release_date',
14         'age_restriction', 'playtime', 'description', 'status'
15     ];
16
17     protected $casts = [
18         'release_date' => 'date',
19     ];
20
21     public function genres()
22     {
23         return $this->belongsToMany(Genre::class, 'movie_genre');
24     }
25
26     public function getFormattedPlaytimeAttribute()
27     {
28         $hours = intval($this->playtime, 60);
29         $minutes = $this->playtime % 60;
30         return sprintf('%dh %dm', $hours, $minutes);
31     }
32 }
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari models **'Movie'** yang digunakan untuk mengelola data film dalam website. Selain itu, terdapat juga penentuan-penentuan model dari data film yang akan digunakan nantinya. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

1. Namespace dan Use Statements

- Bagian ini digunakan untuk mendefinisikan namespace dari model ini sebagai bagian dari direktori **'App\Models'**. Lalu, menggunakan trait

'**HasFactory**' untuk mendukung pembuatan factory. Kemudian, program menggunakan '**Model**' sebagai base class untuk model '**Movie**'.

2. Class Definition dan Traits

↳ Bagian ini digunakan untuk mendefinisikan class '**Movie**' yang memperluas class '**Model**' Eloquent Laravel dan di dalamnya terdapat penggunaan trait '**HasFactory**' yang telah dipanggil sebelumnya.

3. Protected '\$fillable' Property

↳ Properti ini digunakan untuk menentukan kolom yang dapat diisi secara massal (**mass-assignment**) pada tabel '**movies**'. Kolom-kolom ini mencakup '**title**', '**image**', '**rating**', '**production**', '**director**', '**release_date**', '**age_restriction**', '**playtime**', '**description**', dan '**status**'. Hal ini merupakan langkah keamanan untuk mencegah *mass-assignment vulnerability* dari pengisian data film.

4. Protected '\$casts' Property

↳ Properti ini didefinisikan untuk mengonversi atribut '**release_date**' menjadi tipe data '**date**' saat diakses dari model (casting).

5. Relationships on Public function 'genres()'

↳ Bagian ini digunakan untuk mendefinisikan relasi **many-to-many** dengan model '**Genre**'. Metode ini mengembalikan koleksi genre yang terkait dengan film melalui tabel pivot '**movie_genre**'.

6. Accessor on Public function 'getFormattedPlaytimeAttribute()'

↳ Bagian ini digunakan untuk mendefinisikan accessor guna mendapatkan waktu putar ('**playtime**') dalam format jam dan menit. Fungsi ini membagi playtime dalam jam dan menit, lalu mengembalikan string yang diformat seperti ('**2h 30m**').

Dengan demikian, models '**Movie**' digunakan untuk mengelola data film dalam website. Model ini menentukan kolom mana yang dapat diisi, casting kolom '**release_date**' sebagai tanggal, mendefinisikan hubungan many-to-many dengan model '**Genre**', dan memiliki accessor untuk mendapatkan '**playtime**' dalam format yang lebih terbaca oleh pengguna website saya.

Models 'Genre'

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Genre extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = ['name'];
13
14     public function movies()
15     {
16         return $this->belongsToMany(Movie::class, 'movie_genre');
17     }
18 }
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari models 'Genre' yang digunakan untuk mengelola data genre film dalam website. Selain itu, terdapat juga penentuan hubungan dari data genre film yang akan digunakan nantinya. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

1. Namespace dan Use Statements

↪ Bagian ini digunakan untuk mendefinisikan namespace dari model ini sebagai bagian dari direktori 'App\Models'. Lalu, menggunakan trait 'HasFactory' untuk mendukung pembuatan factory. Kemudian, program menggunakan 'Model' sebagai base class untuk model 'Genre'.

2. Class Definition dan Traits

↪ Bagian ini digunakan untuk mendefinisikan class 'Genre' yang memperluas class 'Model' Eloquent Laravel dan di dalamnya terdapat penggunaan trait 'HasFactory' yang telah dipanggil sebelumnya.

3. Protected '\$fillable' Property

↪ Properti ini digunakan untuk menentukan kolom yang dapat diisi secara massal (**mass-assignment**) pada tabel 'genres'. Dalam kasus ini, hanya pada kolom 'name' saja. Hal ini merupakan langkah keamanan untuk mencegah *mass-assignment vulnerability* dari pengisian data film.

4. Relationships on Public function 'movies()'

↪ Bagian ini digunakan untuk mendefinisikan relasi **many-to-many** dengan model 'Movie'. Metode ini mengembalikan koleksi film yang terkait dengan genre melalui tabel pivot 'movie_genre'.

Dengan demikian, models 'Genre' digunakan untuk mengelola data genre film dalam website. Model ini menentukan kolom 'name' yang dapat diisi dan

mendefinisikan hubungan many-to-many dengan model '**Movie**'. Hubungan ini memungkinkan genre untuk dihubungkan dengan banyak film, dan sebaliknya.

Pada dasarnya, Model '**Movie**' dan '**Genre**' bekerja sama untuk mengelola data film dan genre dalam aplikasi berbasis Laravel. Model '**Movie**' menangani atribut film serta relasinya dengan genre, sedangkan model '**Genre**' menangani atribut genre dan relasinya dengan film. Relasi **many-to-many** antara kedua model ini diimplementasikan melalui tabel pivot '**movie_genre**', yang mana memungkinkan pengelolaan genre untuk setiap film secara fleksibel dan efisien. Lalu, pendefinisian Accessor pada model '**Movie**' memberikan tambahan fungsionalitas untuk memformat playtime, meningkatkan kemudahan penggunaan data di tingkat aplikasi.

❖ Factories For '**Movie**' & '**Genre**'

🚧 Factory '**MovieFactory**'



```
1 class MovieFactory extends Factory
2 {
3     protected $model = Movie::class;
4     /**
5      * Define the model's default state.
6      *
7      * @return array<string, mixed>
8      */
9     public function definition(): array
10    {
11        return [
12            'title' => $this->faker->sentence,
13            'image' => $this->faker->imageUrl(),
14            'rating' => $this->faker->randomFloat(1, 0, 10),
15            'production' => $this->faker->company,
16            'director' => $this->faker->name,
17            'release_date' => $this->faker->date,
18            'age_restriction' => $this->faker->randomElement(['G', 'PG', 'PG-13', 'R', 'NC-17']),
19            'playtime' => $this->faker->numberBetween(60, 240),
20            'description' => $this->faker->paragraph,
21            'status' => $this->faker->randomElement(['available', 'unavailable', 'coming_soon']),
22        ];
23    }
24 }
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari factories dari '**MovieFactory.php**' yang digunakan untuk mengonstruksi data dummy pada table '**movies**'. Factory ini nantinya akan digunakan di dalam *seeders* yang sesuai untuk menjalankan pengisian datanya. Inti dari bagian tersebut adalah mengonstruksi pengisian data secara random yang nantinya akan disimpan di dalam database dan ditampilkan pada website. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

1. Namespace dan Import Library

- '**namespace Database\Factories;**': Menentukan namespace dari '**MovieFactory**', yang menunjukkan bahwa factory ini berada di dalam direktori '**Database\Factories**'.

⇒ `'use App\Models\Movie;'`: Mengimpor model **'Movie'** yang akan digunakan dalam factory ini. Hal ini penting agar factory mengetahui model mana yang harus dihubungkan dengan data dummy yang akan dihasilkan.

⇒ `'use Illuminate\Database\Eloquent\Factories\Factory;'`: Mengimpor kelas **'Factory'** dari Laravel Eloquent, yang digunakan sebagai kelas dasar untuk membuat factory.

2. Deklarasi Kelas **'MovieFactory'**

⇒ `'@extends\Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Movie>'`: Komentar ini memberikan informasi bahwa kelas **'MovieFactory'** mewarisi dari **'Factory'** yang dihubungkan dengan model **'Movie'**.

⇒ `'class MovieFactory extends Factory'`: Mendefinisikan kelas **'MovieFactory'** yang mewarisi kelas **'Factory'**.

⇒ `'protected $model = Movie::class;'`: Mendeklarasikan properti **'\$model'** yang menunjukkan model yang akan digunakan oleh factory ini, yaitu **'Movie'**.

3. Method **'definition()'**

⇒ `'public function definition(): array'`: Metode **'definition'** ini mendefinisikan state *default* untuk model **'Movie'**. Metode ini mengembalikan sebuah *array* yang berisikan data dummy untuk setiap atribut pada model.

⇒ `'return [...]';`: Array yang dikembalikan berisi *key-value pair* dimana key merupakan nama atribut pada model **'Movie'** dan value merupakan data dummy yang dihasilkan menggunakan **'Faker'** library.

4. Detail Setiap Atribut di dalam Method **'definition()'**

⇒ `'title' => $this->faker->sentence'`: Menghasilkan judul film secara acak menggunakan **'sentence'** dari Faker.

⇒ `'image' => $this->faker->imageUrl()'`: Menghasilkan URL gambar secara acak menggunakan **'imageUrl'** dari Faker.

⇒ `'rating' => $this->faker->randomFloat(1, 0, 10)'`: Menghasilkan rating acak antara 0 dan 10 dengan satu digit desimal menggunakan **'randomFloat'** dari Faker.

⇒ `'production' => $this->faker->company'`: Menghasilkan nama perusahaan produksi secara acak menggunakan **'company'** dari Faker.

⇒ `'director' => $this->faker->name'`: Menghasilkan nama direktur secara acak menggunakan **'name'** dari Faker.

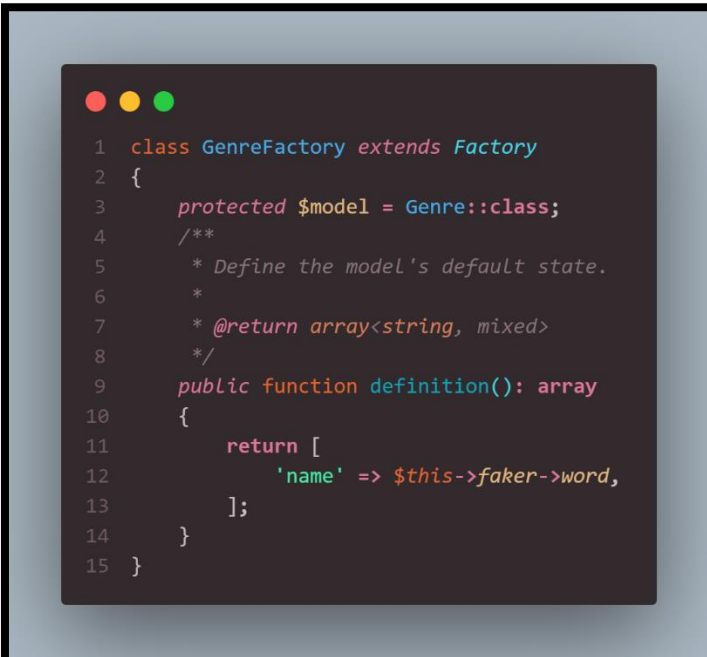
⇒ `'release_date' => $this->faker->date'`: Menghasilkan tanggal rilis secara acak menggunakan **'date'** dari Faker.

⇒ `'age_restriction' => $this->faker->randomElement(['G', 'PG', 'PG-13', 'R', 'NC-17'])'`: Menghasilkan salah satu dari elemen yang diberikan untuk batasan usia menggunakan **'randomElement'** dari Faker.

- ↳ `'playtime' => $this->faker->numberBetween(60, 240)`: Menghasilkan durasi putar secara acak antara 60 hingga 240 menit menggunakan `'numberBetween'` dari Faker.
- ↳ `'description' => $this->faker->paragraph`: Menghasilkan deskripsi secara acak menggunakan `'paragraph'` dari Faker.
- ↳ `'status' => $this->faker->randomElement(['available', 'unavailable', 'coming_soon'])`: Menghasilkan salah satu dari elemen yang diberikan untuk status film menggunakan `'randomElement'` dari Faker.

Dengan demikian, source code Factory **'MovieFactory'** ini digunakan untuk menghasilkan data dummy pada tabel **'movies'** secara otomatis dengan menggunakan **'Faker'**. Data yang dihasilkan mencakup berbagai atribut yang relevan dengan film, seperti judul, gambar, rating, produksi, direktur, tanggal rilis, batasan usia, durasi putar, deskripsi, dan status. Factory ini memudahkan pengujian dan pengembangan dengan menyediakan data yang realistis dan bervariasi tanpa perlu input manual. Factory ini nantinya dapat digunakan di dalam *seeder* untuk mengisi tabel **'movies'** dengan data random.

Factory **'GenreFactory'**



```

1  class GenreFactory extends Factory
2  {
3      protected $model = Genre::class;
4      /**
5       * Define the model's default state.
6       *
7       * @return array<string, mixed>
8       */
9      public function definition(): array
10     {
11         return [
12             'name' => $this->faker->word,
13         ];
14     }
15 }

```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari factories dari **'GenreFactory.php'** yang digunakan untuk mengonstruksi data dummy pada table **'genres'**. Factory ini nantinya akan digunakan di dalam *seeders* yang sesuai untuk menjalankan pengisian datanya. Inti dari bagian tersebut adalah mengonstruksi pengisian data secara random yang nantinya akan disimpan di dalam database dan digunakan oleh setiap data **'movies'** guna

memenuhi relationship yang ada. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

1. Namespace dan Import Library

- ↳ `'namespace Database\Factories;'`: Menyatakan bahwa `'GenreFactory'` berada di dalam namespace `'Database\Factories'`. Namespace ini membantu dalam pengelompokan kode dan menghindari konflik nama.
- ↳ `'use App\Models\Genre;'`: Mengimpor model `'Genre'` yang akan digunakan dalam factory ini. Hal ini penting agar factory mengetahui model mana yang harus dihubungkan dengan data dummy yang akan dihasilkan.
- ↳ `'use Illuminate\Database\Eloquent\Factories\Factory;'`: Mengimpor kelas `'Factory'` dari Eloquent yang merupakan basis dari pembuatan factory di Laravel.

2. Deklarasi Kelas `'GenreFactory'`

- ↳ `'@extends Illuminate\Database\Eloquent\Factories\Factory<App\Models\Genre>'`: Komentar ini memberikan informasi bahwa kelas `'GenreFactory'` mewarisi dari `'Factory'` yang dihubungkan dengan model `'Genre'`.
- ↳ `'class GenreFactory extends Factory'`: Mendefinisikan kelas `'GenreFactory'` yang mewarisi kelas `'Factory'`.
- ↳ `'protected $model = Genre::class;'`: Mendeklarasikan properti `'$model'` yang menunjukkan model yang akan digunakan oleh factory ini, yaitu `'Genre'`.

3. Method `'definition()'`

- ↳ `'public function definition(): array'`: Metode `'definition'` ini mendefinisikan state *default* untuk model `'Genre'`. Metode ini mengembalikan sebuah *array* yang berisi data dummy untuk setiap atribut pada model.
- ↳ `'return [...]';'`: Array yang dikembalikan berisikan *key-value pair* di mana key merupakan nama atribut pada model `'Genre'` dan value merupakan data dummy yang dihasilkan menggunakan `'Faker'` library.
- ↳ Detail atribut di dalam metode `'definition()'` terdapat `'name' => $this->faker->word,'` yang akan menghasilkan satu kata secara acak yang akan digunakan sebagai nama dari genre.

Dengan demikian, source code `'GenreFactory'` merupakan sebuah factory yang digunakan untuk membuat data dummy bagi model `'Genre'` dalam aplikasi Laravel. Dengan mendefinisikan state *default* melalui metode `'definition'`, program dapat menghasilkan data secara acak yang realistis untuk pengujian atau pengisian basis data. Factory ini memanfaatkan library `'Faker'` untuk menghasilkan data acak yang beragam dan realistis. Genre yang dihasilkan oleh factory ini nantinya akan digunakan oleh data setiap `'movies'` guna memenuhi

relationship yang ada, yaitu hubungan antara film dan genre. Proses ini memudahkan pengembangan dan pengujian aplikasi dengan menyediakan data dummy yang bervariasi dan realistis.

❖ Seeders For 'Movies' & 'Genres'

✚ Seeders 'MovieSeeder'



```
1 class MovieSeeder extends Seeder
2 {
3     /**
4      * Run the database seeds.
5      */
6     public function run(): void
7     {
8         // Menambahkan 20 movie baru secara acak
9         Movie::factory()->count(20)->create()->each(function ($movie) {
10             // Setiap movie memiliki 1 hingga 3 genre secara acak
11             $genres = Genre::inRandomOrder()->take(rand(1, 3))->pluck('id');
12             $movie->genres()->attach($genres);
13         });
14     }
15 }
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari seeders 'MovieSeeder.php' yang digunakan untuk mengisi atau menambahkan data 'movies' ke dalam database menggunakan data dummy yang telah dikonstruksi pada factorynya. Berikut akan saya jelaskan perbagiannya dengan lebih rinci:

1. Namespace dan Import Library

- ✚ `'namespace Database\Seeders;':` Menyatakan bahwa 'MovieSeeder' berada di dalam namespace 'Database\Seeders'. Namespace ini membantu dalam pengelompokan kode dan menghindari konflik nama.
- ✚ `'use Illuminate\Database\Console\Seeds\WithoutModelEvents;':` Mengimpor trait 'WithoutModelEvents' yang dapat digunakan untuk menonaktifkan event model selama proses seeding. Meskipun tidak digunakan langsung dalam kode ini, hal ini sering diimpor sebagai bagian dari standar.
- ✚ `'use Illuminate\Database\Seeder;':` Mengimpor kelas 'Seeder' dari Laravel yang menjadi basis dari semua seeder.
- ✚ `'use App\Models\Movie;':` Mengimpor model 'Movie' yang akan digunakan dalam seeder ini.
- ✚ `'use App\Models\Genre;':` Mengimpor model 'Genre' yang akan digunakan untuk mengaitkan genre ke dalam movie.

2. Deklarasi Kelas **'MovieSeeder'**

- ↳ **'class MovieSeeder extends Seeder'**: Mendefinisikan kelas **'MovieSeeder'** yang mewarisi kelas **'Seeder'**. Kelas ini akan digunakan untuk mengisi data ke dalam database.
- ↳ **'public function run(): void'**: Metode **'run'** merupakan metode yang akan dipanggil ketika seeder dijalankan. Metode ini berisi logika untuk mengisi data ke dalam database.

3. Logika Pengisian Data

- ↳ **'Movie::factory()->count(20)->create()'**: Memanggil factory **'Movie'** untuk membuat 20 record movie baru dengan data dummy. Method **'create()'** digunakan untuk menyimpan data tersebut ke dalam database.
- ↳ **'->each(function (\$movie) { ... })'**: Method **'each'** digunakan untuk iterasi setiap **'movie'** yang baru saja dibuat. Dimana, setiap **'movie'** akan diproses di dalam fungsi anonim yang didefinisikan.
- ↳ **'\$genres = Genre::inRandomOrder()->take(rand(1, 3))->pluck('id');'**
 - **'Genre::inRandomOrder()'**: Mengambil genre secara acak dari tabel **'genres'**.
 - **'->take(rand(1, 3))'**: Mengambil antara 1 hingga 3 genre secara acak.
 - **'->pluck('id');'**: Mengambil hanya **'id'** dari genre yang dipilih dalam bentuk koleksi.
- ↳ **'\$movie->genres()->attach(\$genres);'**: Mengaitkan (attach) genre yang dipilih ke dalam **'movie'** yang baru dibuat. Hal ini menambahkan record ke dalam tabel pivot yang menghubungkan **'movies'** dan **'genres'**.

Dengan demikian, source code **'MovieSeeder'** merupakan seeder yang digunakan untuk mengisi data **'movies'** ke dalam database menggunakan data dummy yang dihasilkan oleh **'MovieFactory'**. Seeder ini membuat 20 **'movie'** baru dengan data acak dan kemudian mengaitkan 1 hingga 3 **'genre'** secara acak pula ke setiap movienya. Proses ini memastikan bahwa setiap **'movie'** memiliki beberapa **'genre'** yang terkait, dimana hal ini akan memenuhi hubungan yang ada antara **'movies'** dan **'genres'** dalam database. Seeder ini sangat berguna untuk pengujian dan pengembangan aplikasi website, memungkinkan program untuk memiliki data yang realistis dan beragam dalam database.

Seeders 'GenreSeeder'



```
1 class GenreSeeder extends Seeder
2 {
3     /**
4      * Run the database seeds.
5      */
6     public function run(): void
7     {
8         // Menambahkan 5 genre secara acak
9         Genre::factory()->count(5)->create();
10    }
11 }
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari seeders 'GenreSeeder.php' yang digunakan untuk mengisi atau menambahkan data 'genre' ke dalam database menggunakan data dummy yang telah dikonstruksi pada factorynya. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

1. Namespace dan Import Library

- ↳ `'namespace Database\Seeders;'`: Menyatakan bahwa 'GenreSeeder' berada di dalam namespace 'Database\Seeders'. Namespace ini membantu dalam pengelompokan kode dan menghindari konflik nama.
- ↳ `'use Illuminate\Database\Console\Seeds\WithoutModelEvents;'`: Mengimpor trait 'WithoutModelEvents' yang dapat digunakan untuk menonaktifkan event model selama proses seeding. Meskipun tidak digunakan langsung dalam kode ini, hal ini sering diimpor sebagai bagian dari standar.
- ↳ `'use Illuminate\Database\Seeder;'`: Mengimpor kelas 'Seeder' dari Laravel yang menjadi basis dari semua seeder.
- ↳ `'use App\Models\Genre;'`: Mengimpor model 'Genre' yang akan digunakan dalam seeder ini.

2. Deklarasi Kelas 'GenreSeeder'

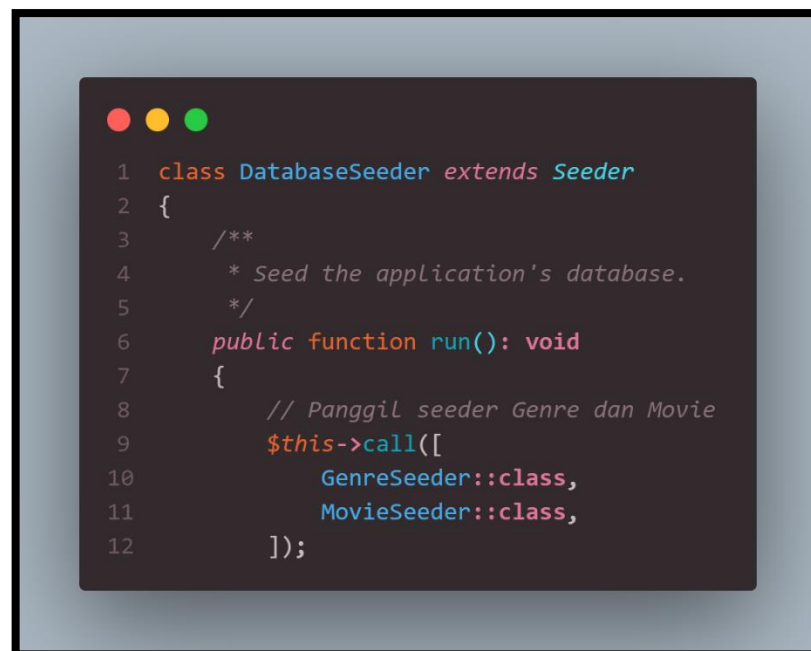
- ↳ `'class GenreSeeder extends Seeder'`: Mendefinisikan kelas `'GenreSeeder'` yang mewarisi kelas `'Seeder'`. Kelas ini akan digunakan untuk mengisi data ke dalam database.
- ↳ `'public function run(): void'`: Metode `'run'` merupakan metode yang akan dipanggil ketika seeder dijalankan. Metode ini berisi logika untuk mengisi data ke dalam database.

3. Logika Pengisian Data

- ↳ `'Genre::factory() ->count(5) ->create()'`: Memanggil factory `'Genre'` untuk membuat 5 record `'genre'` baru dengan data dummy. Method `'create()'` digunakan untuk menyimpan data tersebut ke dalam database.

Dengan demikian, source code `'GenreSeeder'` merupakan seeder yang digunakan untuk mengisi data `'genres'` ke dalam database menggunakan data dummy yang dihasilkan oleh `'GenreFactory'`. Seeder ini membuat 5 `'genre'` baru dengan data acak. Proses ini sangat sederhana namun penting karena memungkinkan program untuk memiliki beberapa genre yang dapat digunakan oleh data `'movies'` guna memenuhi hubungan yang ada antara `'movies'` dan `'genres'` dalam database. Seeder ini memudahkan pengembangan dan pengujian aplikasi dengan menyediakan data dummy yang bervariasi dan realistis dalam tabel `'genres'`.

Seeders 'DatabaseSeeder'

A screenshot of a code editor showing the PHP code for the DatabaseSeeder class. The code is written in a dark-themed editor with a light blue border. The code defines the DatabaseSeeder class, which extends the Seeder class. It includes a comment indicating its purpose is to seed the application's database. The run() method is defined, which calls the create() method of the GenreSeeder and MovieSeeder classes.

```
1 class DatabaseSeeder extends Seeder
2 {
3     /**
4      * Seed the application's database.
5      */
6     public function run(): void
7     {
8         // Panggil seeder Genre dan Movie
9         $this->call([
10             GenreSeeder::class,
11             MovieSeeder::class,
12         ]);
```

Potongan program PHP yang saya lampirkan di atas merupakan bagian dari seeders `'DatabaseSeeder.php'` yang digunakan untuk memanggil seeders yang ada untuk setiap tabel atau model guna memproses pengisian data ke dalam

database menggunakan data dummy yang telah dikonstruksi pada masing-masing factory. Berikut akan saya jelaskan per-bagiannya dengan lebih rinci:

1. Namespace dan Import Library

- ↳ `'namespace Database\Seeders;'`: Menyatakan bahwa `'DatabaseSeeder'` berada di dalam namespace `'Database\Seeders'`. Namespace ini membantu dalam pengelompokan kode dan menghindari konflik nama.
- ↳ `'use Illuminate\Database\Console\Seeds\WithoutModelEvents;'`: Mengimpor trait `'WithoutModelEvents'` yang dapat digunakan untuk menonaktifkan event model selama proses seeding. Meskipun tidak digunakan langsung dalam kode ini, hal ini sering diimpor sebagai bagian dari standar.
- ↳ `'use Illuminate\Database\Seeder;'`: Mengimpor kelas `'Seeder'` dari Laravel yang menjadi basis dari semua seeder.

2. Deklarasi Kelas `'DatabaseSeeder'`

- ↳ `'class DatabaseSeeder extends Seeder'`: Mendefinisikan kelas `'DatabaseSeeder'` yang mewarisi kelas `'Seeder'`. Kelas ini akan digunakan sebagai seeder utama untuk mengisi data ke dalam database.
- ↳ `'public function run(): void'`: Metode `'run'` merupakan metode yang akan dipanggil ketika seeder dijalankan. Metode ini berisi logika untuk mengisi data ke dalam database.

3. Memanggil Seeder Lain

- ↳ `'$this->call([...]);'`: Method `'call'` digunakan untuk memanggil seeders lain dalam aplikasi website. Hal ini memungkinkan program untuk mengorganisasi dan menjalankan beberapa seeders dalam satu tempat.
- ↳ `'GenreSeeder::class, MovieSeeder::class'`: Menentukan seeders yang akan dipanggil, yaitu `'GenreSeeder'` dan `'MovieSeeder'`. Dengan memanggil seeders ini, program memastikan bahwa data untuk genre dan movie akan diisi ke dalam database sesuai dengan logika yang telah didefinisikan di masing-masing seedernya.

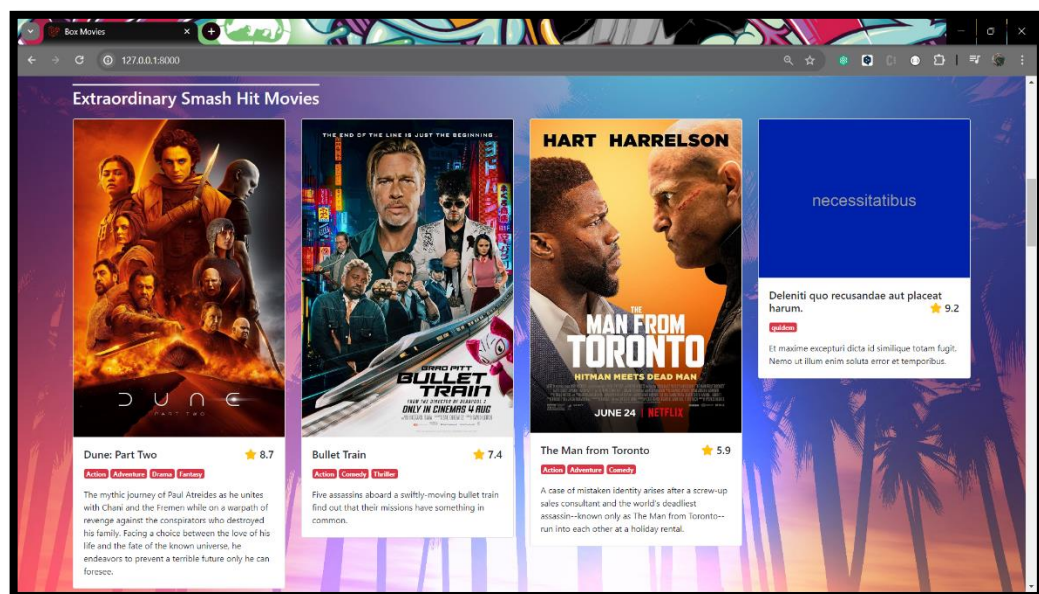
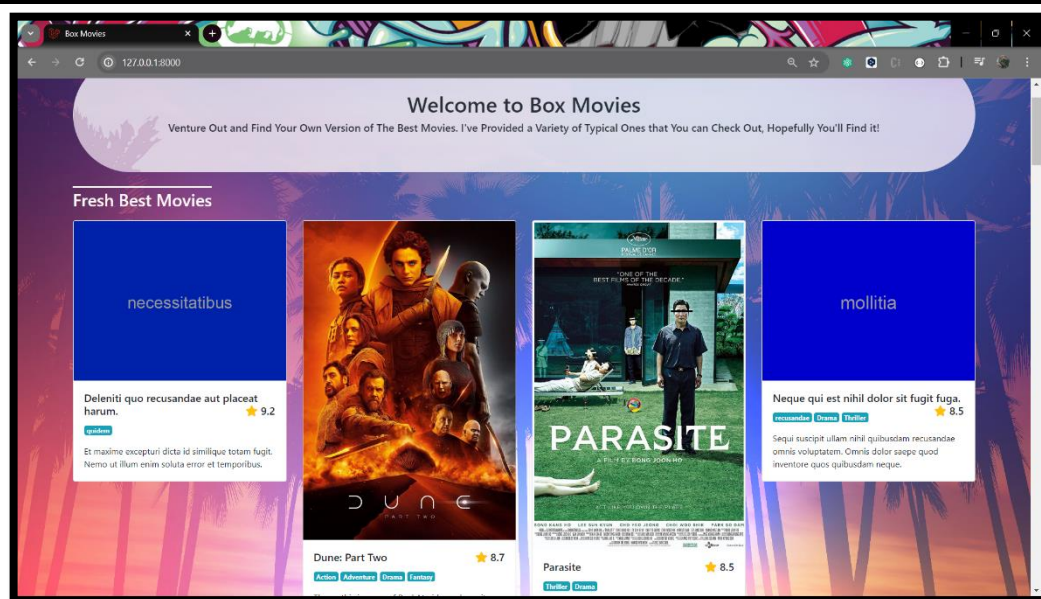
Dengan demikian, source code `'DatabaseSeeder'` merupakan seeder utama yang digunakan untuk memanggil seeders lainnya guna mengisi data ke dalam database menggunakan data dummy yang telah dikonstruksi pada masing-masing factory. Dengan mendefinisikan metode `'run'` yang memanggil seeders `'GenreSeeder'` dan `'MovieSeeder'`, `DatabaseSeeder` memastikan bahwa tabel `'genres'` dan `'movies'` diisi dengan data acak yang realistis. Seeder ini memainkan peran penting dalam pengembangan dan pengujian aplikasi website, memungkinkan pengembang untuk memiliki data lengkap dan terstruktur dalam database dengan sekali eksekusi seeder utama. Proses ini meningkatkan efisiensi dan efektivitas dalam penyiapan lingkungan pengujian.

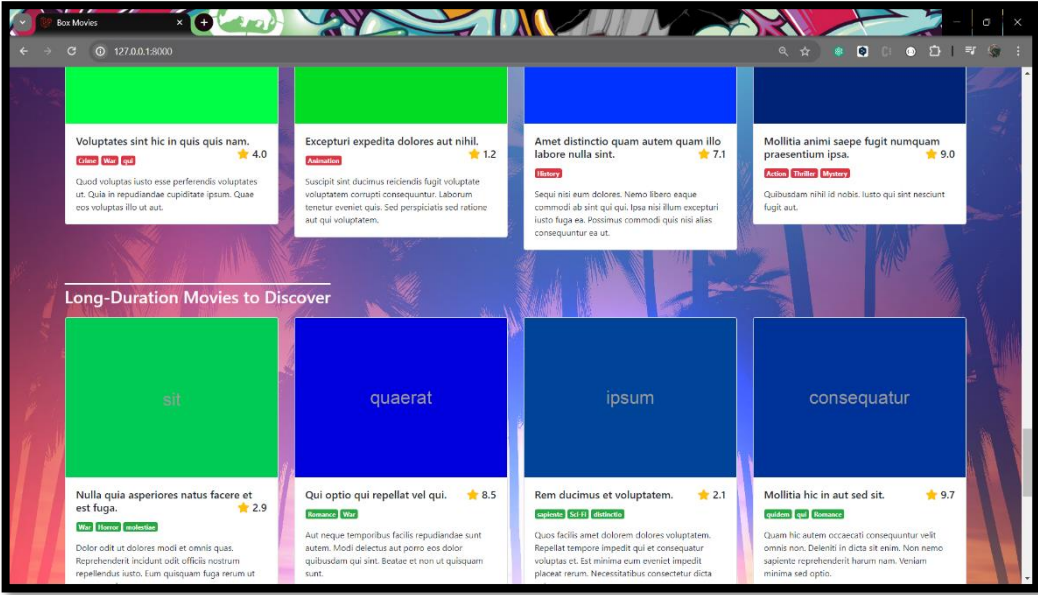
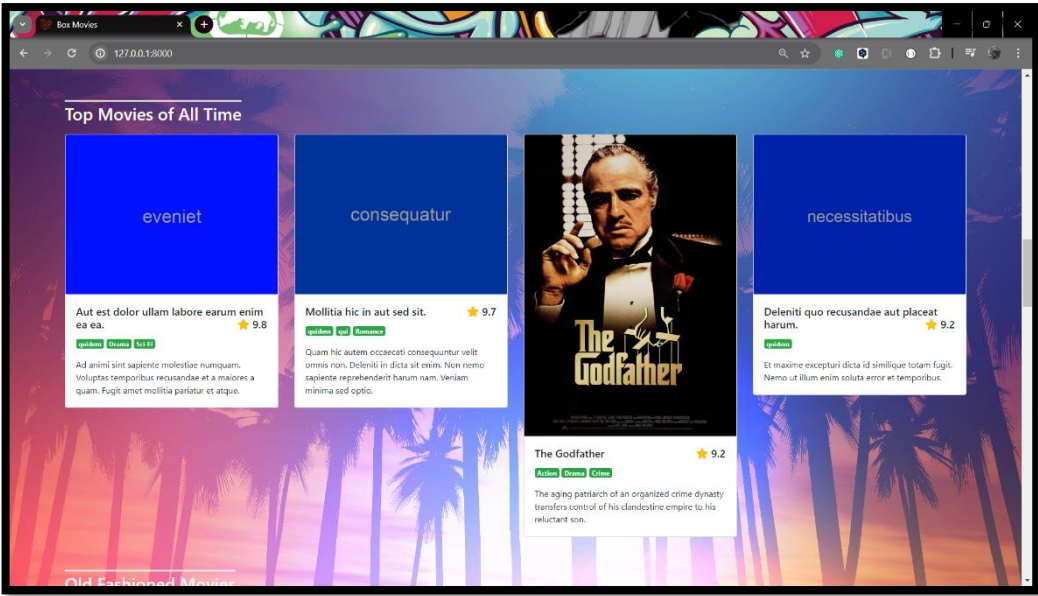
II. Output Program (Laravel Website & PHPMyAdmin Database)

Berikut merupakan output dalam bentuk website localhost dan PHPMyAdmin dari pengembangan program website berbasis Laravel yang saya modifikasi sedemikian rupa pada penugasan minggu ke-12 yang menerapkan Migrations, Seeder, dan Factory untuk mengelola pengisian atau penambahan data secara dummy dalam database. Lampiran yang saya cantumkan berupa media foto (image) saja, apabila hendak menggunakan menggunakan laman saya secara maksimal, dapat diperiksa langsung dari folder ('**box_movies**') yang saya lampirkan pada folder penugasan, terima kasih Mas dan Mbak.

❖ BOX MOVIES HOME SECTION






-WEBSITE OUTPUT-














❖ MOVIES DATA VIEW TABLE

-WEBSITE OUTPUT-

Porro unde ducimus aut dignissimos eos.		6.1	Marquardt and Sons	Garth Keebler	Jan 2017	PG-13	1h 51m	Et omnis impedit nulla minus maxime. Earum consequuntur eos iste ducimus exercitationem sunt ut. Alias ut enim qui delentit odit voluptas ipsa. Perspiciatis quibusdam autem beatae sint illum.	Mystery Horror	Unavailable	Edit Delete
Amet distinctio quam autem quam illo labore nulla sint.		7.1	Cummings-Parker	Madelynn Maggio IV	Feb 2023	R	1h 12m	Sequi nisi eum dolores. Nemo libero eaque commodi ab sint qui qui. Ipsa nisi illum excepturi iusto fuga ea. Possimus commodi quis nisi alias consequuntur ea ut.	History	Unavailable	Edit Delete
Mollitia animi saepe fugit numquam praesentium ipsa.		9.0	Nicolas, Bartell and Hahn	Fletcher Stoltenberg	Dec 1976	NC-17	1h 22m	Quibusdam nihil id nobis. Iusto qui sint nesciunt fugit aut.	Action Thriller Mystery	Unavailable	Edit Delete
Autem eveniet aut quis atque pariatur sit molestiae.		2.7	Mertz Group	D. Mertz MD	Apr 1992	G	2h 57m	Fuga nam quo qui amet. Magna nuncius laudantium inventore fuga aut perant. Sit quisque dolores praesentia.	War Thriller Fantasy	Coming Soon	Edit Delete
Nulla quia asperiores natus		2.9	Kling-Kutch	Barthel	Jul 1992	G	1h 30m	Quis qui ut dolores modi et natus qui. Reprehenderit consequuntur odio officio nostrum.	War Horror	Available	Edit

Rem ducimus et voluptatem.		2.1	Parker, Roy and Rose	Sanford Leffack	Sep 1983	PG-13	3h 24m	Voluptas ut. Sed animae cum conseti impedit placuit rerum. Nonnullatibus consequatur dicta voluptatem.	Comedy Drama Fantasy	Unavailable	Edit Delete
Delentit quo recusandae aut placuit harum.		9.2	Hegmann Inc	Philip Abbott	Aug 2005	R	3h 5m	Et maxime excepturi dicta ut omnisque totam fugit. Nemo ut illum enim soluta error et temporibus.	Comedy	Available	Edit Delete
Ipsam officis id vel consequatur voluptatem praesentium.		3.4	Price, Strach and Klein	Ashley Ward	Sep 2003	NC-17	1h 58m	Maxime ut nihil ut. Inventore et tunc quibusdam voluptate qui. Tempore ipsum optime aut quidem, in corpori et tuncque omnes non.	Action Mystery	Unavailable	Edit Delete
Doloremque cumque qui praesentia aut ipsum saepe ducimus est.		5.9	Robt and Sons	Imad Stoltenberg	May 2015	PG-13	2h 49m	Reprehendus cumque dolorem ipsa reprehenderit officis opto voluptatem. Aperiam iusto sapiente voluptatem accusantium qui corporis enim quibusque. Inventore nesciunt voluptas qui a eum ea.	Action	Available	Edit Delete
Aut est dolor ullam laborum etiam error ea ea.		5.8	Rowe, Koss and Rath	Hector Reynolds	Dec 1978	G	2h 3m	Ad animi qui corporis molestiae voluptatem. Voluptas temporibus nesciunt pergetis maxime a quam. Fugit animi mollitia pariatur et aliquis.	Comedy Drama Sci-Fi	Coming Soon	Edit Delete
Adi dolorem dolorem et repellat tempora.		8.4	Fey Kruma and Schneider	Terence Bartolin	Mar 1978	PG-13	1h 3m	Nulla molestiae non maxime qui et. Architecto voluptas sunt qui mollitia nesciunt. Et nobis enim dolor recusandae praesentium consequatur qui. Ut natus amet nulla aut qui.	Comedy Drama Sci-Fi	Coming Soon	Edit Delete
Qui optio qui repellat vel qui.		6.5	Mills-Johnston	Dr. Margaret Cummings	Jul 2012	PG-13	1h 3m	Aut nesciunt voluptatem qui repellat sunt autem. Reprehenderit enim peris non dolor quibusdam qui corporis et per ut quisque sunt.	Comedy Drama Mystery	Unavailable	Edit Delete
Nesciunt qui et nihil dolor ut fugit fuga.		8.5	Kasperen PSC	Dr. Margaret Cummings	Jul 2012	PG-13	1h 3m	Tempore reprehenderit ut nulla qui nesciunt recusandae nesciunt voluptatem. Corporis dolor nesciunt quibusdam qui corporis et per ut quisque sunt.	Comedy Drama Mystery	Available	Edit Delete
Mollitia eum id eligendi quod praesentium repellat molestiae.		3.3	Kuhner LLC	Dr. Margaret Cummings	Jul 2012	PG-13	1h 3m	Commodi qui nesciunt qui nesciunt recusandae nesciunt voluptatem. Corporis dolor nesciunt quibusdam qui corporis et per ut quisque sunt.	Comedy Drama Mystery	Unavailable	Edit Delete
Quosunt non ducimus peris qui corporis natus.		2.9	Perry Inc	Terence Bartolin	Mar 1978	PG-13	1h 3m	Sed qui nesciunt qui nesciunt recusandae nesciunt voluptatem. Corporis dolor nesciunt quibusdam qui corporis et per ut quisque sunt.	Comedy Drama Sci-Fi	Coming Soon	Edit Delete

MOVIES TABLE
-DATABASE OUTPUT-

GENRE TABLE

-DATABASE OUTPUT-

The screenshot shows the phpMyAdmin web interface in a browser. The address bar indicates the URL is localhost/127.0.0.1/box_movies. The interface is for a database named 'box_movies' and shows a table with the same name. The table has four columns: 'id', 'name', 'created_at', and 'updated_at'. The table contains 41 rows of data, each representing a movie entry. The 'name' column lists various movie genres and titles, such as 'Action', 'Adventure', 'Comedy', 'Thriller', 'Horror', 'Mystery', 'Drama', 'Sci-Fi', 'Crime', 'Romance', 'Family', 'Biography', 'War', 'History', 'Fantasy', 'Animation', 'Dystopia', 'Sci-Fi', 'Reincarnation', 'Sapientia', 'Quidem', 'Molierstae', 'Eum', 'Iuga', and 'Voluptatem'. The 'created_at' and 'updated_at' columns show timestamps for each entry.