Web Framework Programming

Topik 3:
Database Connection, Migration & Seeding

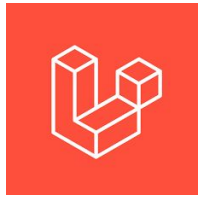
WEEK 03

Informatics Engineering
Universitas Surabaya

OUTLINE

- Connecting Laravel with MySQL
- File .env
- Migration Concept
- Migration Syntax & Case study
- Seeding Concept
- Seeding Syntax & Case study

Native DB in Laravel



As Default, Laravel is able to connect several databases:

- MariaDB
- MySQL
- PostgreSQL
- SQLite
- SQL Server

Default Configuration

```php
'default' => env('DB_CONNECTION', 'mysql'),
'connections' => [

    'sqlite' => [
        'driver' => 'sqlite',
        'url' => env('DATABASE_URL'),
        'database' => env('DB_DATABASE', database_path('database.sqlite')),
        'prefix' => '',
        'foreign_key_constraints' => env('DB_FOREIGN_KEYS', true),
    ],

    'mysql' => [
        'driver' => 'mysql',
        'url' => env('DATABASE_URL'),
        'host' => env('DB_HOST', '127.0.0.1'),
        'port' => env('DB_PORT', '3306'),
        'database' => env('DB_DATABASE', 'forge'),
        'username' => env('DB_USERNAME', 'forge'),
        'password' => env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'prefix_indexes' => true,
```

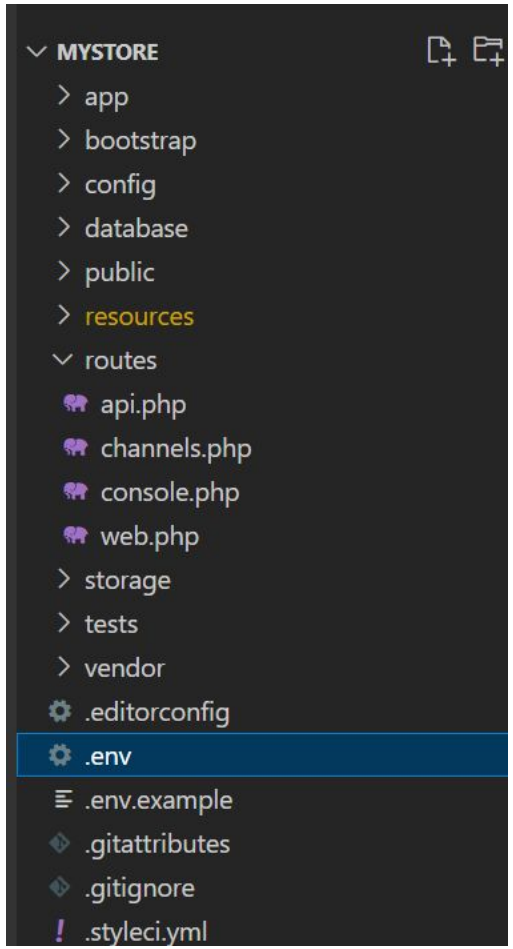
- Database access setting is located at <u>config/database.php</u>
- Please look at this configuration in your new laravel project

env (param1, param2) is syntax for colleting value from variable base on .env (environment) file

For information, .env file should be place on root of your Laravel Project

param1 = KEY
param2 = default_value

File .env




```
1  APP_NAME=Laravel
2  APP_ENV=local
3  APP_KEY=base64:yz3wif1+YdR90t5
4  APP_DEBUG=true
5  APP_URL=http://localhost
6
7  LOG_CHANNEL=stack
8  LOG_DEPRECATIONS_CHANNEL=null
9  LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
```

This is the configuration to connect the database. If you use your local database server, it just need to set up line 12 to 14

Migration

- *Build Database Table using migration*

Migrations

Laravel has a “magic” tool to build, modify, and delete database table structures. It is called **Migrations**.

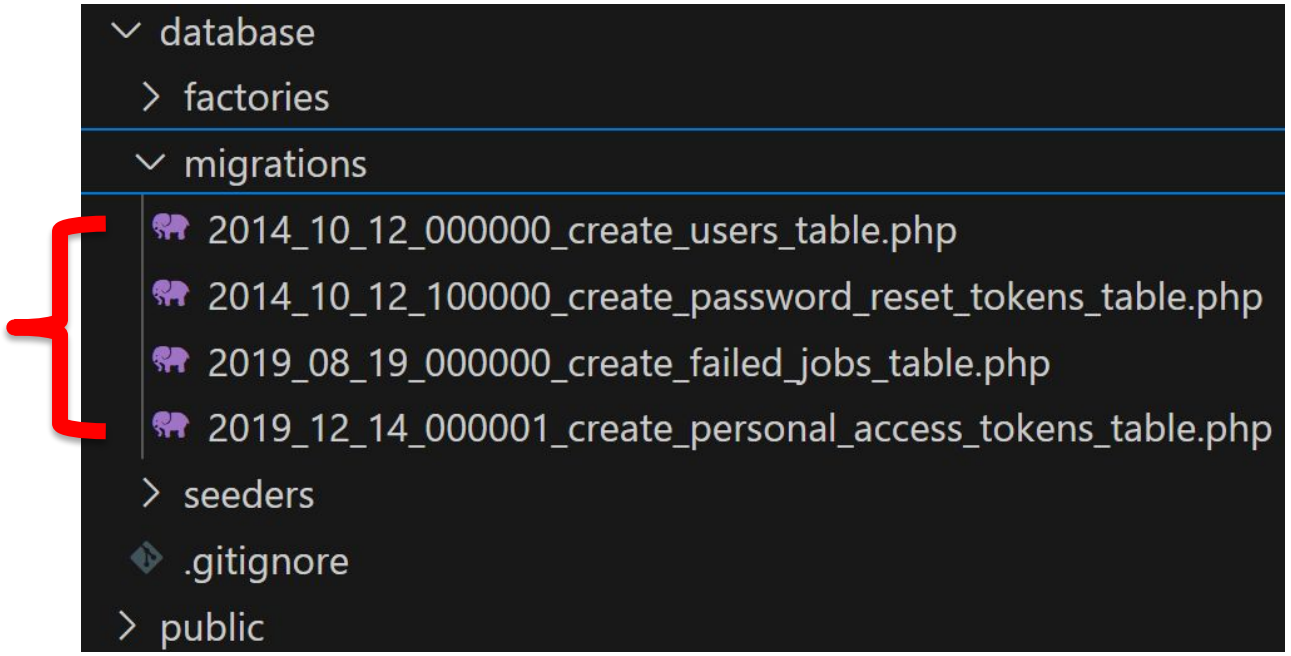
Migrations are like version control for your database, allowing your team to modify and share the application's database schema. Migrations are typically paired with Laravel's schema builder to build your application's database schema.

from: https://laravel.com/docs/10.x/migrations#introduction

File Location

Migration files is placed on /database/migrations

By default, Laravel provides 4 file migrations that will create a User table, Password Reset, Failed Jobs table and Personal Access Token. The “user” table will be used to store the user of the application in the authentication process. Failed Jobs table will be used to store the failed activity. Personal access tokens is for APIs development in Laravel.



DDL

- A migration file **represent a DDL** (Data Definition Language) statement to create or modify a database schema/table.
- A migration file is built with the **Artisan** command: **php artisan make:migration *namefile***

- All artisan file (which is not yet executed) are run with the **Artisan** command: **php artisan migrate**

Based on: https://laravel.com/docs/10.x/migrations

Characteristics

- Migrations file will **run/execute only once** after the programmer “runs” migration.
- If it need a modification to the existing schema, you should have to create a new migrations file and do a modifying syntax with “Update table” syntaxes in Migration files.
- If you need to go back to the previous version of the database structure, use the rollback command Artisan: **php artisan migrate:rollback**
- Here, the versioning feature is important
- the migrations file name has a special format to represent the action in it. In the previous example: create_users_table,
  create => build new,
  _users =>table name,
  _table => the entity type table

Naming Conventions

The naming convention for table name use the plural noun.

- for example: if you have a “user” data, then in the database the table will named: ‘users’
- The background of this naming convention is came from the Eloquent Model, which will be explained later.

https://laravel.com/docs/10.x/eloquent#eloquent-model-conventions

Function in Migration Files

A migration file has two function : up() and down().

- **up()** is using for executing the forward structure modification
- **down()** is using for undoing the structure modification (backward)

```php

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
};
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }
```

UP Method

up() in the previous figure will create new table named ‘users’ which have this attributes/column:

a. id (PK, Unsigned BIG INT)
b. Name (varchar/string)
c. Email (varchar/string, unique)
d. Email_verified_at (null, Timestamp)
e. Password (varchar/string)
f. Remember_token (varchar/string)
g. Created_at & updated_at (timestamp)

More about creating table:
https://laravel.com/docs/10.x/migrations#creating-tables
More about the columns data types
https://laravel.com/docs/10.x/migrations#columns

Down Method

Down() always opposites with up() function.
If the up() function creates a table, the down() function will delete the table structure.

More about dropping tables
https://laravel.com/docs/10.x/migrations#renaming-and-dropping-tables
More about the columns data types
https://laravel.com/docs/10.x/migrations#columns

There are 2 syntax for deleting table :

```
Schema::drop('users');

Schema::dropIfExists('users');
```

The second syntax will avoid error if there is no related table to be drop in the database.

Foreign Key

The syntax for foreign key:
https://laravel.com/docs/10.x/migrations#foreign-key-constraints

```
Schema::table('posts', function (Blueprint $table) {
    $table->unsignedBigInteger('user_id');

    $table->foreign('user_id')->references('id')->on('users');
});
```

The explanation of the above syntax:
Change the structure of table 'posts' by **adding a new column user_id,** with type Unsigned BigInteger, and make the **'user_id' column a Foreign Key** that relates to column 'id' in the '**users**' table

Case Study

Case

- Self Ordering System has various foods and their categories
- This system have 2 entities:
  - Foods to store the name of foods, description, Nutrition Facts, Price
  - Categories to store the general type of categories such as: Appetizer, Main Course, Snacks, Dessert, Coffee, Non-Coffee, Healthy Juice.

Exercise

Please construct this database

Example

1. Foods :
   - **Name** : Nasi Merah dengan Ayam Panggang Kecap & Tumis Kangkung
   - **Description** : Nikmati hidangan sehat dan lezat dengan Nasi Merah yang kaya serat, dipadukan dengan Ayam Panggang Kecap yang manis gurih dan Tumis Kangkung yang segar. Kombinasi sempurna untuk santapan yang mengenyangkan dan bergizi.
   - **Price:** Rp. 35.000,-
   - **Nutrition facts** :

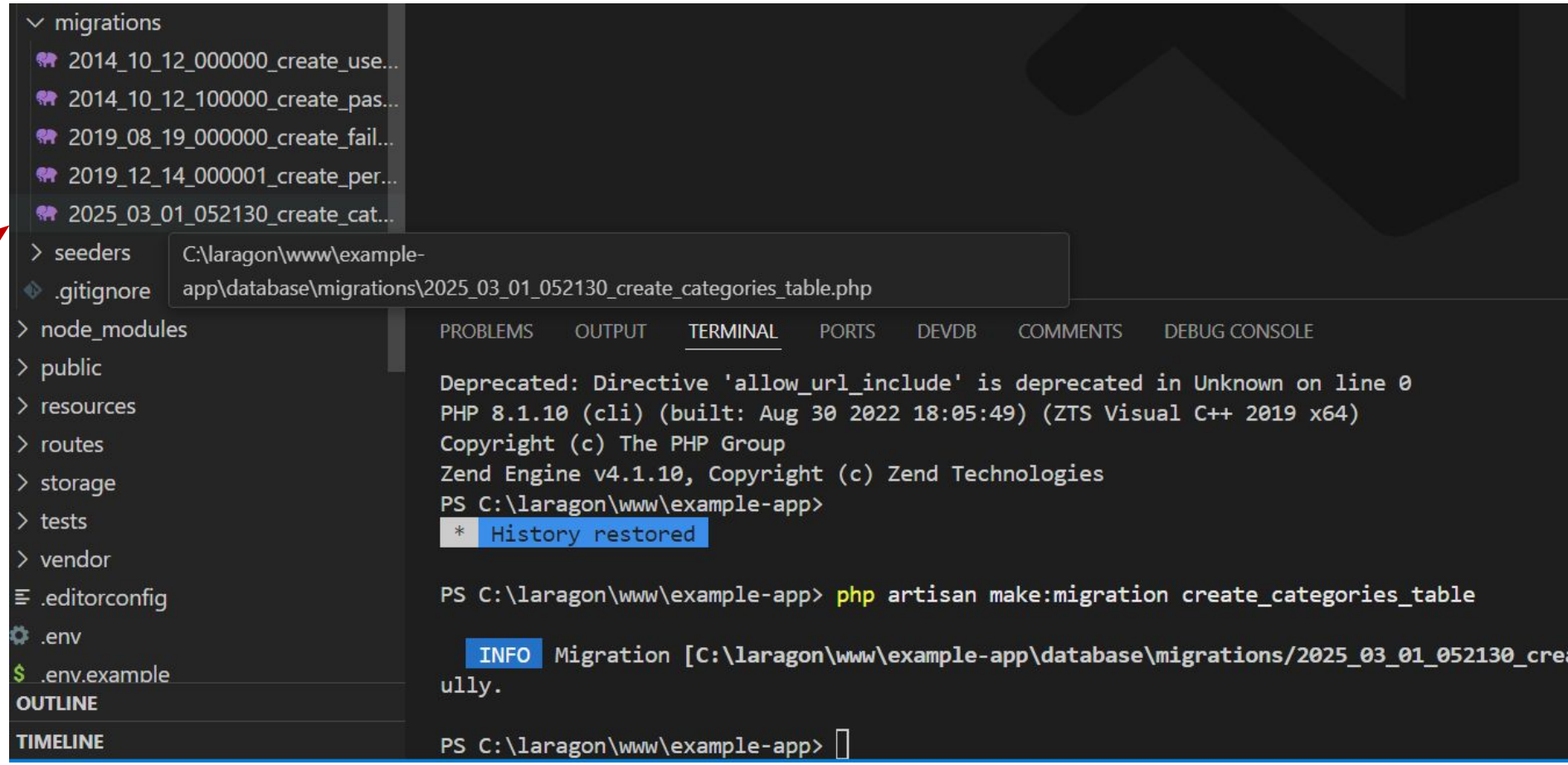
     Kalori: 400-550 kkal
     Protein: 30-40 gram
     Lemak: 15-25 gram
     Karbohidrat: 50-70 gram
     Serat: 5-8 gram

2. Categories : Appetizer, Main Course, Snacks, Dessert, Coffee, Non-Coffee, Healthy Juice

Practice #1

Migration for types

Result



```
PS C:\laragon\www\example-app> php artisan make:migration create_foods_table

  INFO  Migration [C:\laragon\www\example-app\database\migrations/2025_03_01_052230_
```

- database
  - factories
  - migrations
    - 2014_10_12_000000_create_users_table.php
    - 2014_10_12_100000_create_password_reset_tokens_table.php
    - 2019_08_19_000000_create_failed_jobs_table.php
    - 2019_12_14_000001_create_personal_access_tokens_table.php
    - 2025_03_01_052130_create_categories_table.php
    - 2025_03_01_052230_create_foods_table.php

Create_categories_table.php

Laravel will **automatically add up() and down()** function and fill with some code. We just need to add (or remove) another fields which not already there.

2025_03_01_052130_create_categories_table.php

database > migrations > 2025_03_01_052130_create_categories_table.php > ...

```php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    Codeium: Refactor | Explain | X
    public function up(): void
    {
        Schema::create(table: 'categories', callback: function (Blueprint $table): void {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    Codeium: Refactor | Explain | X
    public function down(): void
    {
        Schema::dropIfExists(table: 'categories');
    }
};

```

Create_places_table.php

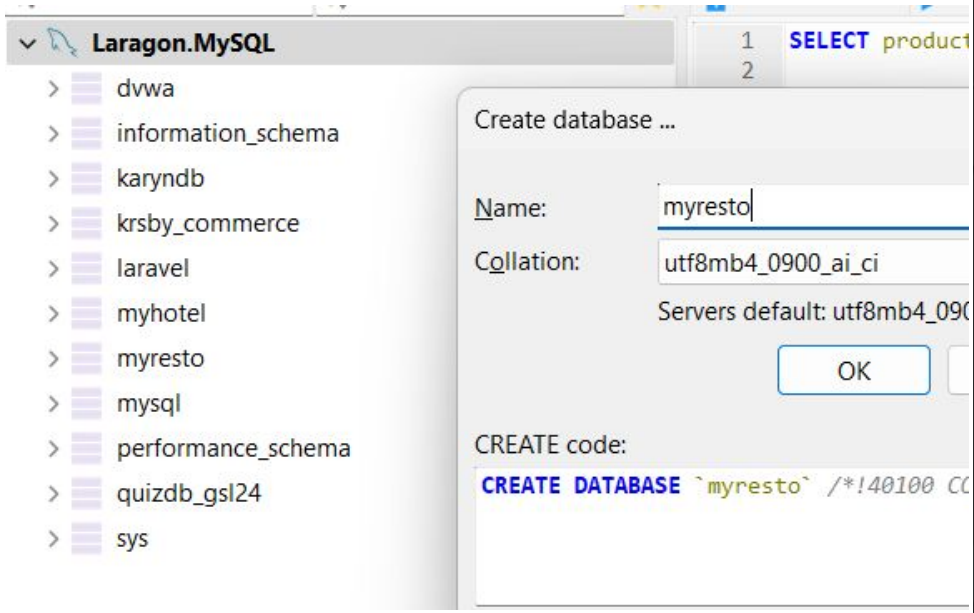
Result

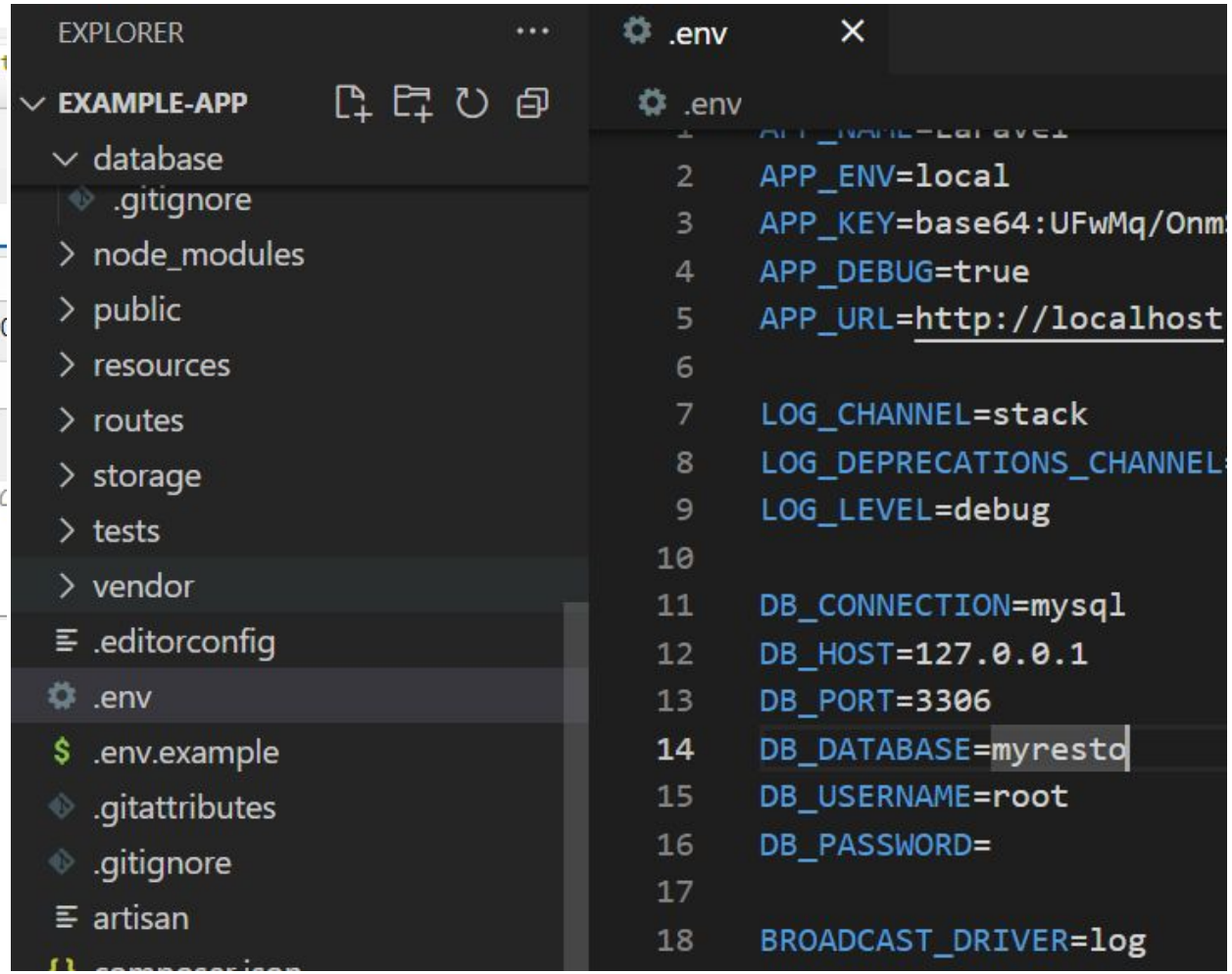
2025_03_01_052130_create_categories_table.php | 2025_03_01_052230_create_foods_table.php ×

database > migrations > 2025_03_01_052230_create_foods_table.php > ...

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    Codeium: Refactor | Explain | ×
    public function up(): void
    {
        Schema::create(table: 'foods', callback: function (Blueprint $table): void {
            $table->id();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    Codeium: Refactor | Explain | ×
    public function down(): void
    {
        Schema::dropIfExists(table: 'foods');
    }
};
```

Make sure your Connection database before Run Migration





Before running it, make sure that your setting in .env already refer correct database

Migration Result

Run **php artisan migrate** and see result

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=myresto
DB_USERNAME=root
DB_PASSWORD=

BROADCAST_DRIVER=log
```

```
PS C:\laragon\www\example-app> php artisan migrate

  INFO  Preparing database.

  Creating migration table ..................................... 35ms DONE

  INFO  Running migrations.

  2014_10_12_000000_create_users_table ......................... 46ms DONE
  2014_10_12_100000_create_password_reset_tokens_table ......... 18ms DONE
  2019_08_19_000000_create_failed_jobs_table ................... 38ms DONE
  2019_12_14_000001_create_personal_access_tokens_table ........ 62ms DONE
  2025_03_01_052130_create_categories_table .................... 18ms DONE
  2025_03_01_052230_create_foods_table ......................... 16ms DONE

PS C:\laragon\www\example-app>
```

Database filter | Table filter | Host: 127.0.0.1 | Database: myresto | Query

- Laragon.MySQL
  - dvwa
  - information_schema
  - karyndb
  - krsby_commerce
  - laravel
  - myhotel
  - **myresto** 112.0 KiB
    - categories 16.0 KiB
    - failed_jobs 16.0 KiB
    - foods 16.0 KiB
    - migrations 16.0 KiB
    - password_reset_tokens 16.0 KiB

Name	Rows	Size	Crea
categories	0	16.0 KiB	202
failed_jobs	0	16.0 KiB	202
foods	0	16.0 KiB	202
migrations	0	16.0 KiB	202
password_rese...	0	16.0 KiB	202
personal_acces...	0	16.0 KiB	202
users	0	16.0 KiB	202

Host: 127.0.0.1 | Database: myresto | Table: categories | Data | Query

Basic | Options | Indexes (1) | Foreign keys (0) | Check constraints (0) | Partitions | CRI

Name: categories

Comment:

Columns: Add | Remove | Up | Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Defau
1	id	BIGINT	20	☑	☐	☐	AUT
2	created_at	TIMESTAMP		☐	☑	☐	NUL
3	updated_at	TIMESTAMP		☐	☑	☐	NUL

Host: 127.0.0.1 | Database: myresto | Table: foods | Data | Query

Basic | Options | Indexes (1) | Foreign keys (0) | Check constraints (0) | Partitions | CREATE code | ALT

Name: foods

Comment:

Columns: Add | Remove | Up | Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Co
1	id	BIGINT	20	☑	☐	☐	AUTO_INCREME...	
2	created_at	TIMESTAMP		☐	☑	☐	NULL	
3	updated_at	TIMESTAMP		☐	☑	☐	NULL	

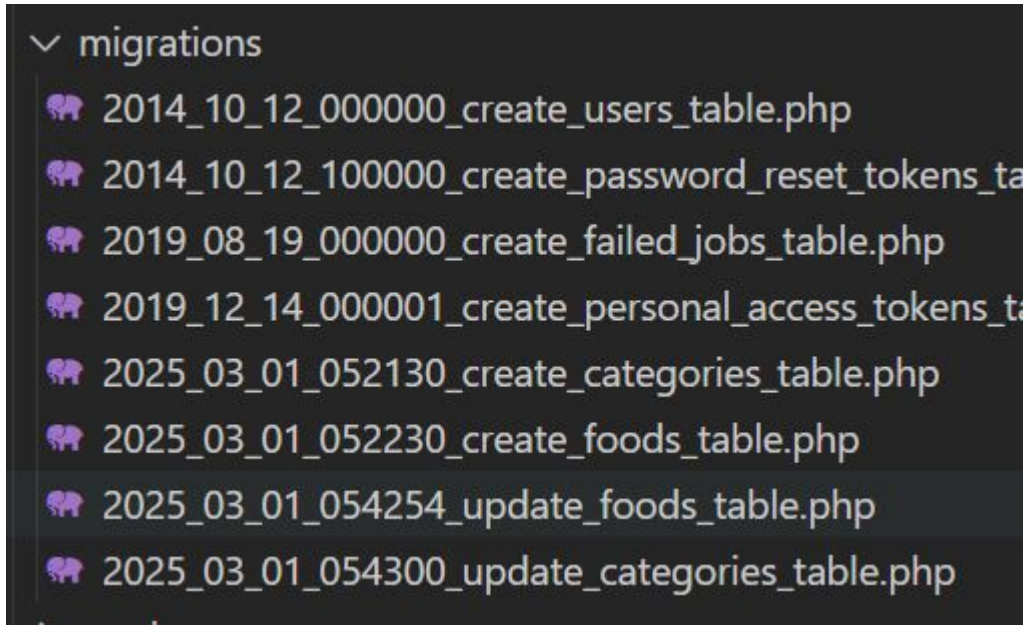
Practice #2

- How can we improve with specific coloumn?

New Migration for update attributes

```
PS C:\laragon\www\example-app> php artisan make:migration update_foods_table

  INFO  Migration [C:\laragon\www\example-app\database\migrations/2025_03_01_054254_update_foods_table.php] created successf
ully.

PS C:\laragon\www\example-app> php artisan make:migration update_categories_table

  INFO  Migration [C:\laragon\www\example-app\database\migrations/2025_03_01_054300_update_categories_table.php] created suc
cessfully.
```

We will try to update our two tables with a new attributes



[REMEMBER]

One migration file is only one time execution. If you want to update something after execution, you can make a new file with update statement

New Migration for update attributes

https://laravel.com/docs/10.x/migrations#modifying-columns
https://laravel.com/docs/10.x/migrations#available-column-types

Two new files are still **empty.**

We need to fulfill the correct syntax with "Creating Column" , "Available Column Types", "Dropping Column"

```
7   return new class extends Migration
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         //
15     }
16
17     /**
18      * Reverse the migrations.
19      */
20     public function down(): void
21     {
22         //
23     }
24 };
```

Available Column Types

The schema builder blueprint offers a variety of methods that correspond to the different types of columns you can add to your database tables. Each of the available methods are listed in the table below:

bigIncrements	jsonb	string
bigInteger	lineString	text
binary	longText	timeTz
boolean	macAddress	time
char	mediumIncrements	timestampTz
dateTimeTz	mediumInteger	timestamp
dateTime	mediumText	timestampsTz
date	morphs	timestamps
decimal	multiLineString	tinyIncrements

Update your attributes (Files)

Foods :

- **id**
- **name**
- **nutrition_fact**
- **description**
- **price**
- **category_id (FK)**
- Created_at

  kapan data dibuat
- Updated_at

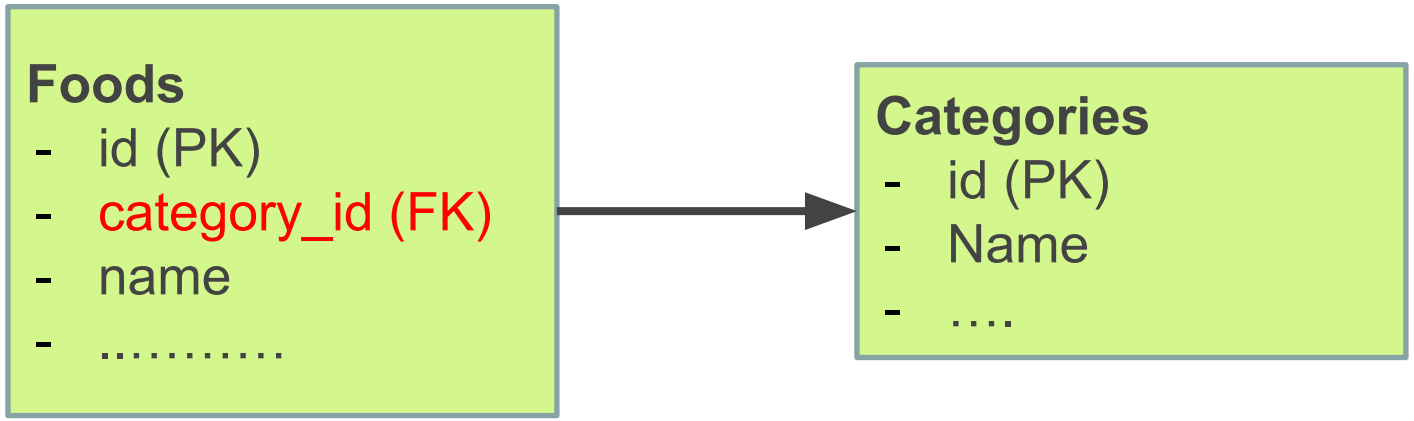
  kapan data diupdate terakhir

Categories :

- **id**
- **name**

Update for Relationship

- In many cases, we need to add foreign key in the table. In our system, we need to add ‘category_id’ into ‘food’ table. This represents that **one place** has **<u>only</u> one category**, and **one category** has **<u>many</u> foods**.



- Exercise: *“How to modify table, add new column, and create relation with a column from another table”*

Foods Migration Update

```php
public function up(): void
{
    Schema::table('foods', function (Blueprint $table) {
        $table->string('name', 150);
        $table->text('nutrition_fact');
        $table->text('description');
        $table->float('price', 8, 2);

        $table->unsignedBigInteger('category_id');
        $table->foreign('category_id')->references('id')->on('categories');
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    //
    Schema::table('foods', function (Blueprint $table) {
        $table->dropColumn(['name', 'nutrition_fact', 'description','price']);

        $table->dropForeign(['category_id']);
    });
```

Please write and check with Laravel Documentation & Your Case Study

Category Migration Update

```php
public function up(): void
{
    Schema::table('categories', function (Blueprint $table) {
        $table->string('name', 50);
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::table('categories', function (Blueprint $table) {
        $table->dropColumn(['name']);
    });
}
```

Please write and check with Laravel Documentation & Your Case Study

Run migration after completed

```
PS C:\laragon\www\example-app> php artisan migrate

  INFO  Running migrations.

  2025_03_01_054254_update_foods_table ..................................... 108ms DONE
  2025_03_01_054300_update_categories_table ................................ 26ms DONE

PS C:\laragon\www\example-app>
```

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned
1	id	BIGINT	20	☑
2	created_at	TIMESTAMP		☐
3	updated_at	TIMESTAMP		☐
4	name	VARCHAR	150	☐
5	nutrition_fact	TEXT		☐
6	description	TEXT		☐
7	price	DOUBLE	8,2	☐
8	category_id	BIGINT	20	☑

Basic | Options | Indexes (2) | Foreign keys (1) | Check constraints (0) | Partitions

Add / Remove / Clear

Key name	Columns	Reference table	Foreign col...	On UPDATE	On DELETE
foods_categ...	category_id	myresto.categ...	id	NO ACTION	NO ACTION

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Def
1	id	BIGINT	20	☑	☐	☐	AU
2	created_at	TIMESTAMP		☐	☑	☐	NU
3	updated_at	TIMESTAMP		☐	☑	☐	NU
4	name	VARCHAR	150	☐	☐	☐	No
5	nutrition_fact	TEXT		☐	☐	☐	No
6	description	TEXT		☐	☐	☐	No
7	price	DOUBLE	8,2	☐	☐	☐	No
8	category_id	BIGINT	20	☑	☐	☐	No

Seeding

Seeding

- Seeding in Laravel means to fill in and modify data in the table
- File Seeding use for initial data of an application. It is often used as dummy data to demonstrate the application features.
  Seeding files are located at ‘database/seeds’
- Seeding is not have a versioning mechanism like in the migration
- Seeding file listed in DatabaseSeeder.php

https://laravel.com/docs/10.x/seeding

DatabaseSeeder.php

DatabaseSeeder.php is the mainClass of Seeding implementation.

Laravel will read the content of statemen run() and run it sequentially. Seeder file can be executed over and over.

```
5   // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6   use Illuminate\Database\Seeder;
7
    0 references | 0 implementations
8   class DatabaseSeeder extends Seeder
9   {
10      /**
11       * Seed the application's database.
12       */
        0 references | 0 overrides
13      public function run(): void
14      {
15          // \App\Models\User::factory(10)->create();
16
17          // \App\Models\User::factory()->create([
18          //     'name' => 'Test User',
19          //     'email' => 'test@example.com',
20          // ]);
21      }
22  }
23
```

Fill table with seeder

There are **2 ways** to fill table with data. (1) Query Builder; (2) Eloquent Model Factory.

In this section we use **Query Builder** with insert() method and **Faker** (https://fakerphp.org/)

The steps are

1. Create Seeder File

- A seeder file represent a table. If you have 4 tables than you will have maximum 4 seeder file
- Syntax in command Artisan to create seeder: **php artisan make:seeder UserSeeder**
- UserSeeder represent a filling for table User.

```
C:\xampp\htdocs\example-app>php artisan make:seeder UserSeeder

  INFO  Seeder [C:\xampp\htdocs\example-app\database\seeders\UserSeeder.php] created successfully.
```

```
8   class UserSeeder extends Seeder
9   {
10      /**
11       * Run the database seeds.
12       */
13      public function run(): void
14      {
15          //
16      }
17  }
18
```

1. Create Seeder File (2)

Create seeder for foods and categories

```
PS C:\laragon\www\example-app> php artisan make:seeder FoodSeeder

  INFO  Seeder [C:\laragon\www\example-app\database\seeders\FoodSeeder.php] created successfully.

PS C:\laragon\www\example-app> php artisan make:seeder CategorySeeder

  INFO  Seeder [C:\laragon\www\example-app\database\seeders\CategorySeeder.php] created successfully.
```

```
database
  factories
  migrations
  seeders
    CategorySeeder.php
    DatabaseSeeder.php
    FoodSeeder.php
    UserSeeder.php
```

2. Using Query Builder

Based on https://laravel.com/docs/10.x/seeding#writing-seeders

- Query Builder documentation can be found at https://laravel.com/docs/10.x/queries#insert-statements
- Each seeder can be written with this query builder syntax.

```
public function run()
{
    DB::table('users')->insert([
        'name' => Str::random(10),
        'email' => Str::random(10).'@gmail.com',
        'password' => Hash::make('password'),
    ]);
}
```

Practice #4

- 15 minutes

Fill CategorySeeder

```php
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;

class CategorySeeder extends Seeder
{
    public function run(): void
    {
        DB::table('categories')->insert([
            ['name' => 'Appetizer'],
            ['name' => 'Main Course'],
            ['name' => 'Snack'],
            ['name' => 'Dessert'],
            ['name' => 'Coffee'],
            ['name' => 'Non Coffee'],
            ['name' => 'Healthy Juice'],
        ]);
    }
}
```

Fill Food

```php
class FoodSeeder extends Seeder
{
    public function run(): void
    {
        DB::table('foods')->insert([
            [
                'name'=> 'Nasi Merah dengan Ayam Panggang Kecap & Tumis Kangkung',
                'nutrition_fact'=>'Kalori: 400-550 kkal
                            Protein: 30-40 gram
                            Lemak: 15-25 gram
                            Karbohidrat: 50-70 gram
                            Serat: 5-8 gram',
                'description'=>'Nikmati hidangan sehat dan lezat
                         dengan Nasi Merah yang kaya serat, dipadukan dengan Ayam Panggang
                'price'=>35000
            ],
            [
                'name'=> 'Nasi Hitam dan Tumis Ca Kailan',
                'nutrition_fact'=>'Kalori: 400-550 kkal
                            Protein: 30-40 gram
                            Lemak: 15-25 gram
                            Karbohidrat: 50-70 gram
                            Serat: 5-8 gram',
                'description'=>'Nikmati hidangan sehat dan lezat
                         dengan Nasi Hitam yang kaya serat.',
                'price'=>30000
            ],
        ]);
    }
}
```

Add 3 items in food

```php
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;
```

Do not forget to add library DB & str for random on the top of class seeding

Check your UserFactory (use Faker)

```php
    public function definition(): array
    {
        return [
            'name' => fake()->name(),
            'email' => fake()->unique()->safeEmail(),
            'email_verified_at' => now(),
            'password' => static::$password ??= Hash::make('password'),
            'remember_token' => Str::random(10),
        ];
    }

    /**
     * Indicate that the model's email address should be unverified.
     */
    public function unverified(): static
    {
        return $this->state(fn (array $attributes) => [
            'email_verified_at' => null,
        ]);
    }
}
```

Modify DatabaseSeeder

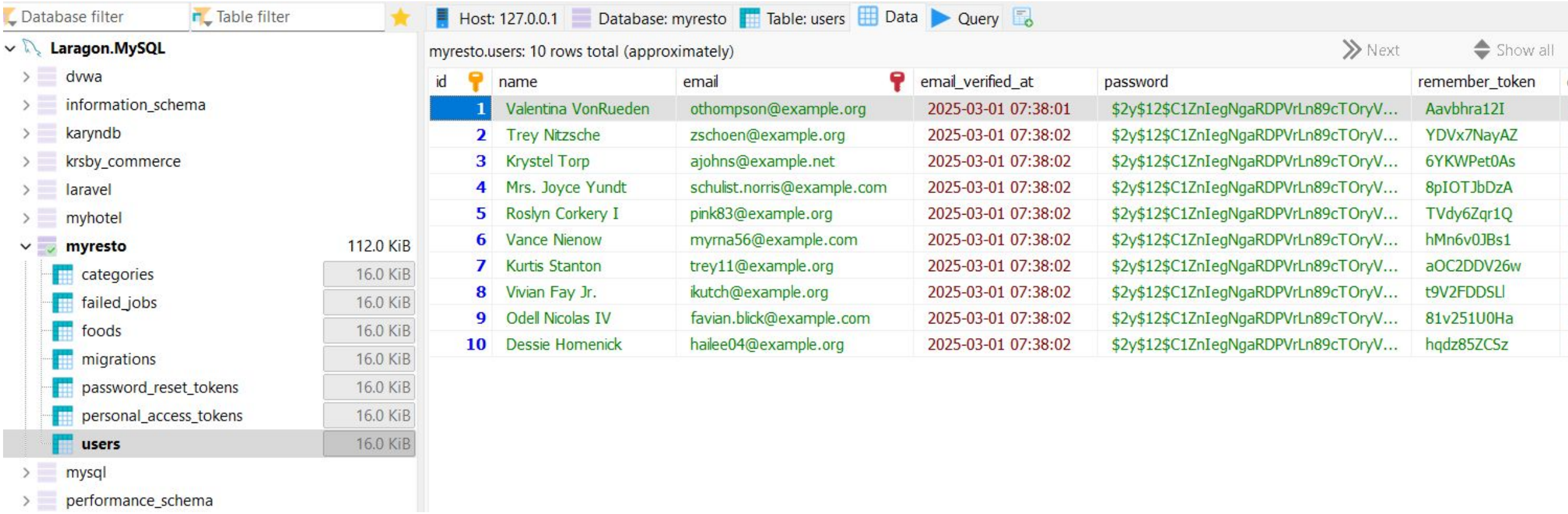
```php
class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        User::factory(10)->create();

        $this->call([
            CategorySeeder::class,
            FoodSeeder::class,
        ]);

        // \App\Models\User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);
    }
}
```

Run seeder

```
PS C:\laragon\www\example-app> php artisan db:seed

  INFO  Seeding database.

  Database\Seeders\CategorySeeder ..................................... RUNNING
  Database\Seeders\CategorySeeder ..................................... 5 ms DONE

  Database\Seeders\FoodSeeder ......................................... RUNNING
  Database\Seeders\FoodSeeder ......................................... 4 ms DONE

PS C:\laragon\www\example-app>
```

Open folder in new window (ctrl + click)

Result User



id	name	email	email_verified_at	password	remember_token
1	Valentina VonRueden	othompson@example.org	2025-03-01 07:38:01	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	Aavbhra12I
2	Trey Nitzsche	zschoen@example.org	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	YDVx7NayAZ
3	Krystel Torp	ajohns@example.net	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	6YKWPet0As
4	Mrs. Joyce Yundt	schulist.norris@example.com	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	8pIOTJbDzA
5	Roslyn Corkery I	pink83@example.org	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	TVdy6Zqr1Q
6	Vance Nienow	myrna56@example.com	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	hMn6v0JBs1
7	Kurtis Stanton	trey11@example.org	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	aOC2DDV26w
8	Vivian Fay Jr.	ikutch@example.org	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	t9V2FDDSLl
9	Odell Nicolas IV	favian.blick@example.com	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	81v251U0Ha
10	Dessie Homenick	hailee04@example.org	2025-03-01 07:38:02	$2y$12$C1ZnIegNgaRDPVrLn89cTOryV...	hqdz85ZCSz

Result Foods and Categories

Database filter | Table filter | Host: 127.0.0.1 | Database: myresto | Table: categories | Da

- Laragon.MySQL
  - dvwa
  - information_schema
  - karyndb
  - krsby_commerce
  - laravel
  - myhotel
  - **myresto** 112.0 KiB
    - **categories** 16.0 KiB
    - failed_jobs 16.0 KiB
    - foods 16.0 KiB
    - migrations 16.0 KiB
    - password_reset_tokens 16.0 KiB

myresto.categories: 7 rows total (approximately)

id	created_at	updated_at	name
1	(NULL)	(NULL)	Appetizer
2	(NULL)	(NULL)	Main Course
3	(NULL)	(NULL)	Snack
4	(NULL)	(NULL)	Dessert
5	(NULL)	(NULL)	Coffee
6	(NULL)	(NULL)	Non Coffee
7	(NULL)	(NULL)	Healthy Juice

Database filter | Table filter | Host: 127.0.0.1 | Database: myresto | Table: foods | Data | Query

- Laragon.MySQL
  - dvwa
  - information_schema
  - karyndb
  - krsby_commerce
  - laravel
  - myhotel
  - **myresto** 112.0 KiB
    - categories 16.0 KiB
    - failed_jobs 16.0 KiB
    - **foods** 16.0 KiB
    - migrations 16.0 KiB
    - password_reset_tokens 16.0 KiB
    - personal_access_tokens 16.0 KiB
    - users 16.0 KiB
  - mysql
  - performance_schema

myresto.foods: 2 rows total (approximately) | Next | Show all | Sorting | Columns (8/8) | Filt

id	created_at	updated_at	name	nutrition_fact		description		price	category_id
1	(NULL)	(NULL)	Nasi Merah dengan Ayam Panggang Ke...	Kalori: 400-550 kkal	...	Nikmati hidangan sehat dan lezat	...	35,000.0	2
2	(NULL)	(NULL)	Nasi Hitam dan Tumis Ca Kailan	Kalori: 400-550 kkal	...	Nikmati hidangan sehat dan lezat	...	30,000.0	2

Another Issues

```
1   vendor\laravel\framework\src\Illuminate\Database\Connection.php:587
    PDOException::("SQLSTATE[HY000]: General error: 1364 Field 'category_id' doesn't have a default value")

2   vendor\laravel\framework\src\Illuminate\Database\Connection.php:587
    PDOStatement::execute()
```

Faker

https://fakerphp.org/formatters/

Exercise

Exercise for Project Milestones

- Please make 3 tables: (a) Customers, (b) Transactions, (c) Payment
- Make: (a) Customer migration, seeder, (b) Transaction migration seeder, (C) Payment migration seeder
- Two entities will support:

1. Ordering Systems
2. Show Catalogs
3. Payment Mechanism

Thank You