**Task 1:**

Create a new migration file to add a new table named "categories" to the database. The table should have the following columns:

id (primary key, auto-increment)

name (string)

created\_at (timestamp)

updated\_at (timestamp)

**ANS:**

To create a new migration file to add a new table named "categories" to the database, you can use the following command:

**php artisan make:migration create\_categories\_table --create=categories**

This command will generate a new migration file in the database/migrations directory. Open the generated migration file and add the following code:

**<?php**

**use Illuminate\Database\Migrations\Migration;**

**use Illuminate\Database\Schema\Blueprint;**

**use Illuminate\Support\Facades\Schema;**

**class CreateCategoriesTable extends Migration**

**{**

**public function up()**

**{**

**Schema::create('categories', function (Blueprint $table) {**

**$table->increments('id');**

**$table->string('name');**

**$table->timestamp('created\_at')->nullable();**

**$table->timestamp('updated\_at')->nullable();**

**});**

**}**

**public function down()**

**{**

**Schema::dropIfExists('categories');**

**}**

**}**

Save the file and then run the migration using the following command:

**php artisan migrate**

**Task 2:**

Create a new model named "Category" associated with the "categories" table. Define the necessary properties and relationships.

ANS:

To create a new model named "Category" associated with the "categories" table, you can use the following command:

**php artisan make:model Category**

This command will generate a new model file named "Category.php" in the app/Models directory. Open the generated model file and add the necessary properties and relationships:

**<?php**

**namespace App\Models;**

**use Illuminate\Database\Eloquent\Model;**

**class Category extends Model**

**{**

**protected $fillable = ['name'];**

**public function posts()**

**{**

**return $this->hasMany(Post::class);**

**}**

**}**

**Task 3:**

Write a migration file to add a foreign key constraint to the "posts" table. The foreign key should reference the "categories" table on the "category\_id" column.

ANS:

To write a migration file to add a foreign key constraint to the "posts" table, you can create a new migration using the following command:

**php artisan make:migration add\_category\_id\_to\_posts\_table –table=posts**

Open the generated migration file and add the following code:

**<?php**

**use Illuminate\Database\Migrations\Migration;**

**use Illuminate\Database\Schema\Blueprint;**

**use Illuminate\Support\Facades\Schema;**

**class AddCategoryIdToPostsTable extends Migration**

**{**

**public function up()**

**{**

**Schema::table('posts', function (Blueprint $table) {**

**$table->unsignedBigInteger('category\_id')->after('id')->nullable();**

**$table->foreign('category\_id')**

**->references('id')**

**->on('categories')**

**->onDelete('restrict');**

**});**

**}**

**public function down()**

**{**

**Schema::table('posts', function (Blueprint $table) {**

**$table->dropForeign(['category\_id']);**

**$table->dropColumn('category\_id');**

**});**

**}**

**}**

Save the file and then run the migration using the following command:

**php artisan migrate**

**Task 4:**

Create a relationship between the "Post" and "Category" models. A post belongs to a category, and a category can have multiple posts.

ANS:

**To create a relationship between the "Post" and "Category" models, open the "Post" model file (app/Models/Post.php) and add the following code:**

**<?php**

**namespace App\Models;**

**use Illuminate\Database\Eloquent\Model;**

**class Post extends Model**

**{**

**protected $fillable = ['title', 'content', 'category\_id'];**

**public function category()**

**{**

**return $this->belongsTo(Category::class);**

**}**

**}**

**Task 5:**

Write a query using Eloquent ORM to retrieve all posts along with their associated categories. Make sure to eager load the categories to optimize the query.

ANS:

To retrieve all posts along with their associated categories using Eloquent ORM, you can use the following code:

**$posts = Post::with('category')->get();**

**foreach ($posts as $post) {**

**echo "Post: {$post->title}\n";**

**echo "Category: {$post->category->name}\n";**

**echo "\n";**

**}**

**Task 6:**

Implement a method in the "Post" model to get the total number of posts belonging to a specific category. The method should accept the category ID as a parameter and return the count.

ANS:

To implement a method in the "Post" model to get the total number of posts belonging to a specific category, open the "Post" model file (app/Models/Post.php) and add the following code:

**<?php**

**namespace App\Models;**

**use Illuminate\Database\Eloquent\Model;**

**class Post extends Model**

**{**

**protected $fillable = ['title', 'content', 'category\_id'];**

**public function category()**

**{**

**return $this->belongsTo(Category::class);**

**}**

**public static function countPostsByCategory($categoryId)**

**{**

**return self::where('category\_id', $categoryId)->count();**

**}**

**}**

**Task 7:**

Create a new route in the web.php file to handle the following URL pattern: "/posts/{id}/delete". Implement the corresponding controller method to delete a post by its ID. Soft delete should be used.

ANS:

To create a new route in the web.php file to handle the URL pattern "/posts/{id}/delete" and implement the corresponding controller method to delete a post by its ID using soft delete, you can add the following code to the routes/web.php file:

**use App\Http\Controllers\PostController;**

**Route::get('/posts/{id}/delete', [PostController::class, 'delete'])->name('posts.delete');**

This code defines a GET route that maps to the delete method of the PostController class. Make sure to replace PostController with the actual name of your controller class.

In your PostController, you can define the delete method as follows:

**<?php**

**namespace App\Http\Controllers;**

**use App\Models\Post;**

**class PostController extends Controller**

**{**

**public function delete($id)**

**{**

**$post = Post::findOrFail($id);**

**$post->delete();**

**// Redirect or return a response**

**}**

**}**

**Task 8:**

Implement a method in the "Post" model to get all posts that have been soft deleted. The method should return a collection of soft deleted posts.

ANS:

To implement a method in the "Post" model to get all posts that have been soft deleted, open the "Post" model file (app/Models/Post.php) and add the following code:

**<?php**

**namespace App\Models;**

**use Illuminate\Database\Eloquent\Model;**

**use Illuminate\Database\Eloquent\SoftDeletes;**

**class Post extends Model**

**{**

**use SoftDeletes;**

**protected $fillable = ['title', 'content', 'category\_id'];**

**public function category()**

**{**

**return $this->belongsTo(Category::class);**

**}**

**public static function getSoftDeletedPosts()**

**{**

**return self::onlyTrashed()->get();**

**}**

**}**

**Task 9:**

Write a Blade template to display all posts and their associated categories. Use a loop to iterate over the posts and display their details.

ANS:

To write a Blade template to display all posts and their associated categories, create a new Blade template file (e.g., posts.blade.php) and add the following code:

**@foreach ($posts as $post)**

**<h2>{{ $post->title }}</h2>**

**<p>{{ $post->content }}</p>**

**<p>Category: {{ $post->category->name }}</p>**

**@endforeach**

**Task 10:**

Create a new route in the web.php file to handle the following URL pattern: "/categories/{id}/posts". Implement the corresponding controller method to retrieve all posts belonging to a specific category. The category ID should be passed as a parameter to the method.

ANS:

To create a new route in the web.php file to handle the URL pattern "/categories/{id}/posts" and implement the corresponding controller method to retrieve all posts belonging to a specific category, you can add the following code to the routes/web.php file:

**use App\Http\Controllers\CategoryController;**

**Route::get('/categories/{id}/posts', [CategoryController::class, 'getPosts'])->name('categories.posts');**

This code defines a GET route that maps to the getPosts method of the CategoryController class. Replace CategoryController with the actual name of your controller class.

In your CategoryController, you can define the getPosts method as follows:

**<?php**

**namespace App\Http\Controllers;**

**use App\Models\Category;**

**class CategoryController extends Controller**

**{**

**public function getPosts($id)**

**{**

**$category = Category::findOrFail($id);**

**$posts = $category->posts;**

**// Return or pass $posts to a view**

**}**

**}**

**Task 11:**

Implement a method in the "Category" model to get the latest post associated with the category. The method should return the post object.

ANS:

To implement a method in the "Category" model to get the latest post associated with the category, open the "Category" model file (app/Models/Category.php) and add the following code:

**<?php**

**namespace App\Models;**

**use Illuminate\Database\Eloquent\Model;**

**class Category extends Model**

**{**

**protected $fillable = ['name'];**

**public function posts()**

**{**

**return $this->hasMany(Post::class);**

**}**

**public function latestPost()**

**{**

**return $this->hasOne(Post::class)->latest();**

**}**

**}**

**Task 12:**

Write a Blade template to display the latest post for each category. Use a loop to iterate over the categories and display the post details.

ANS:

To write a Blade template to display the latest post for each category, create a new Blade template file (e.g., latest\_posts.blade.php) and add the following code:

**@foreach ($categories as $category)**

**<h2>{{ $category->name }}</h2>**

**<p>Latest Post: {{ $category->latestPost->title }}</p>**

**<p>{{ $category->latestPost->content }}</p>**

**@endforeach**