Task 1: Creating migration files for "categories" tables

Since a foreign key constraint to the "posts" table should be added and foreign key should reference the "categories" table on the "category_id" column, need to create migration file for the "posts" table along with "categories" table.

To create migration files for the "categories" and "posts" tables, you can use the make:migration Artisan command. Run the following commands in your terminal:

For "categories" table:

php artisan make:migration create_categories_table --create=categories

For "posts" table:

php artisan make:migration create_posts_table --create=posts

These commands will generate two migration files: create_categories_table.php and create_posts_table.php. Open each migration file and update their contents as follows:

For "categories" table:

```
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->id();
            $table->string('name');
            $table->timestamps();
        });
```

```
    }

    public function down()
    {
        Schema::dropIfExists('categories');
    }
}
```

For "posts" table:

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

class CreatePostsTable extends Migration
{
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->timestamps();
        });
    }

    public function down()
```

```
    {
        Schema::dropIfExists('posts');
    }
}
```

To run the migrations and create the tables in the database, need to run the following command:

php artisan migrate

Task 2: Creating the "Category" model.

Here it is needed to create "Post" model too along with "Category" model.

To create the "Category" and "Post" models in Laravel, the make:model Artisan command can be used. Run the following commands in the terminal:

For "Category" model:

php artisan make:model Category

For "Post" model:

php artisan make:model Post

These commands will generate two model files: Category.php and Post.php in the app directory. Open each generated model file and define the necessary properties and relationships.

In Category.php:

```
namespace App;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    protected $fillable = ['name'];
```

```
    public function posts()
    {
        return $this->hasMany(Post::class);
    }
}
```

In Post.php:

```
namespace App;
use Illuminate\Database\Eloquent\Model;
class Post extends Model
{
    protected $fillable = ['title'];
    public function category()
    {
        return $this->belongsTo(Category::class);
    }
}
```

Task 3: Adding a foreign key constraint to the "posts" table

To add a foreign key constraint to the "posts" table need to create a new migration file. Need to run the following command in the terminal:

```
php artisan make:migration add_category_id_to_posts --table=posts
```

This command will generate a new migration file named something like 2023_07_03_123456_add_category_id_to_posts.php. Open the migration file and update its contents as follows:

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
```

```
use Illuminate\Support\Facades\Schema;

class AddCategoryIdToPosts extends Migration
{
    public function up()
    {
        Schema::table('posts', function (Blueprint $table) {
            $table->foreignId('category_id')->constrained('categories');
        });
    }

    public function down()
    {
        Schema::table('posts', function (Blueprint $table) {
            $table->dropForeign(['category_id']);
            $table->dropColumn('category_id');
        });
    }
}
```

To run the migration and apply the foreign key constraint, need to run the following command:

```
php artisan migrate
```

Task 4: Establishing the relationship between the "Post" and "Category" models

Open the "Post" model file (app/Post.php) and add the following code to define the relationship between the "Post" and "Category" models:

```
namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

   protected $fillable = ['title'];


   public function category()

   {

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

   }

}
```

Task 5: Querying posts with their associated categories using Eloquent ORM

To retrieve all posts along with their associated categories using Eloquent ORM, you can use the with method to eager load the "category" relationship. Here's an example query:

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

Task 6: Implementing a method to get the total number of posts belonging to a specific category.

Open the "Post" model file (app/Post.php) and add the following method:

```
public static function getCountByCategoryId($categoryId)

{

   return self::where('category_id', $categoryId)->count();

}
```

Task 7: Creating a route and controller method to delete a post.

In the routes/web.php file, add the following route definition:

```
Route::delete('/posts/{id}/delete', 'PostController@delete')->name('posts.delete');
```

In the app/Http/Controllers/PostController.php file, add the following method:

```
public function delete($id)
{
    $post = Post::findOrFail($id);
    $post->delete();
    return redirect()->route('posts.index')->with('success', 'Post deleted successfully.');
}
```

Task 8: Implementing a method to get all soft deleted posts.

Open the "Post" model file (app/Post.php) and add the following method:

```
public static function getSoftDeleted()
{
    return self::onlyTrashed()->get();
}
```

Task 9: Creating a Blade template to display 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>Category: {{ $post->category->name }}</p>
    <p>Created at: {{ $post->created_at }}</p>
    <p>Updated at: {{ $post->updated_at }}</p>
@endforeach
```

Task 10: Creating a route and controller method to retrieve posts belonging to a specific category.

In the routes/web.php file, add the following route definition:

```
Route::get('/categories/{id}/posts', 'CategoryController@getPosts')->name('categories.posts');
```

In the app/Http/Controllers/CategoryController.php file, add the following method:

```
public function getPosts($id)
{
    $category = Category::findOrFail($id);
    $posts = $category->posts;

    return view('categories.posts', compact('category', 'posts'));
}
```

Task 11: Implementing a method to get the latest post associated with a category

Open the "Category" model file (app/Category.php) and add the following method:

```
public function latestPost()
{
    return $this->hasOne(Post::class)->latest();
}
```

Task 12: Creating 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>Created at: {{ $category->latestPost->created_at }}</p>
    <p>Updated at: {{ $category->latestPost->updated_at }}</p>
@endforeach
```