

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)

CLI: `php artisan make:migration create_categories_table --create`

Migration Image file:




```
1 public function up(): void
2 {
3     Schema::create('categories', function (Blueprint $table) {
4         $table->id();
5         $table->string('name',100);
6         $table->timestamp('created_at')->useCurrent();
7         $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
8     });
9 }
10 }
```

Task 2:

Create a new model named "Category" associated with the "categories" table. Define the necessary properties and relationships. Defining Relationship: hasMany(Post::class)

CLI:

```
php artisan make:model Category
```



```
1  <?php
2
3  namespace App\Models;
4
5  use App\Models\Post;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Factories\HasFactory;
8
9  class Category extends Model
10 {
11     use HasFactory;
12     protected $fillable = ['name'];
13     public function posts(){
14         return $this->hasMany(Post::class);
15     }
16
17     public function allLatestPost(){
18
19         return $this->posts()->latest('updated_at')->first();
20
21     }
22
23 }
24
```

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.

CLI: php artisan make:migration create_posts_table --create



```
1 public function up(): void
2 {
3     Schema::create('posts', function (Blueprint $table) {
4         $table->id();
5         $table->string('name');
6         $table->text('description')->nullable();
7         $table->timestamp('created_at')->useCurrent();
8         $table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
9         $table->softDeletes();
10        $table->foreignId('category_id')->constrained('categories')
11            ->restrictOnDelete()
12            ->cascadeOnUpdate();
13    });
14 }
```

Task 4:

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

```
1 namespace App\Models;
2
3 use App\Models\Category;
4 use Illuminate\Database\Eloquent\Model;
5 use Illuminate\Database\Eloquent\SoftDeletes;
6 use Illuminate\Database\Eloquent\Factories\HasFactory;
7
8 class Post extends Model
9 {
10     use HasFactory, SoftDeletes;
11
12     protected $fillable = ['name', 'description', 'category_id'];
13
14     protected $dates = ['deleted_at'];
15     public function category(){
16         return $this->belongsTo(Category::class);
17     }
18
19     public static function categoryWisePostCount($categoryId){
20         return self::where('category_id', $categoryId)->count();
21     }
22
23     public static function softDeletedRows(){
24         return self::onlyTrashed()->get();
25     }
26
27 }
28
```

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.



```
1 public function allPost(){
2     $data['posts'] = Post::with('category')->get();
3
4     return view('pages.home',$data);
5
6 }
```

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.



```
1 public static function categoryWisePostCount($categoryId){
2     return self::where('category_id', $categoryId)->count();
3 }
```

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.



```
1 Route::get('posts/{id}/delete', 'delete');
```

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.



```
1 public static function softDeletedRows(){  
2     return self::onlyTrashed()->get();  
3 }
```

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.

[Home](#)[Post](#) [Categories](#)

All Post With Category

Web Title**Web**

Web Description

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.



```
1 Route::get('/categories/{id}/posts', 'categoryWisePost');
```

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.

```
1 public function allLatestPost(){
2
3     return $this->posts()->latest('updated_at')->first();
4
5 }
```

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.

[Home](#)[Post](#) [Categories](#)

Latest Post For Each Category

Web

Web Title

Web Description