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:

-1) php artisan make:migration create_categories_table --create=categories

-2)
```
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');
    }
}
```

-3) php artisan migrate

Task 2:

Create a new model named "Category" associated with the "categories" table.

Define the necessary properties and relationships.

Ans:-

1) php artisan make:model Category

2)

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

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

   // Relationships
   // Example: A category has many posts
   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:-

1). php artisan make:migration add_category_id_to_posts_table --table=posts

2). 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->foreignId('category_id')->constrained('categories');
      });
   }

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

3). 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:

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
   // ...

   // Define the relationship
   public function category()
```

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

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    // ...

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

```
$post = Post::find(1);
$category = $post->category; // Retrieves the associated category
```

```
$category = Category::find(1);
$posts = $category->posts; // Retrieves all posts belonging to the category
```

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:-

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

```
foreach ($posts as $post) {
    echo 'Post Title: ' . $post->title;
    echo 'Post Category: ' . $post->category->name;
}
```

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:

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


$categoryCount = Post::countPostsByCategory($categoryId);
```

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:-

```
use App\Http\Controllers\PostController;

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



namespace App\Http\Controllers;

use App\Models\Post;
use Illuminate\Http\Request;

class PostController extends Controller
{
   public function delete(Request $request, $id)
   {
      $post = Post::find($id);
```

```
        if ($post) {
            $post->delete();

            // Optionally, you can redirect the user to a specific route or display a success message
            return redirect()->route('posts.index')->with('success', 'Post deleted successfully.');
        }

        // Handle the case when the post is not found
        return redirect()->route('posts.index')->with('error', 'Post not found.');
    }}
```

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.

```
use Illuminate\Database\Eloquent\SoftDeletes;


use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Post extends Model
{
    use SoftDeletes;

    // ...
}
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Post extends Model
{
    use SoftDeletes;

    // ...

    public static function getSoftDeletedPosts()
    {
        return static::withTrashed()->whereNotNull('deleted_at')->get();
    }
```

```
    // ...
}


$softDeletedPosts = Post::getSoftDeletedPosts();
```

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.

```
@foreach ($posts as $post)
    <h2>{{ $post->title }}</h2>
    <p>{{ $post->content }}</p>

    <h4>Categories:</h4>
    <ul>
        @foreach ($post->categories as $category)
            <li>{{ $category->name }}</li>
        @endforeach
    </ul>

    <hr>
@endforeach
```

```
namespace App\Http\Controllers;

use App\Models\Post;

class PostController extends Controller
{
    public function index()
    {
        $posts = Post::with('categories')->get();
```

```
        return view('posts', compact('posts'));
    }
}
```

```
use App\Http\Controllers\PostController;

Route::get('/posts', [PostController::class, 'index'])->name('posts.index');
```

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.

```
use App\Http\Controllers\CategoryController;

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

```
namespace App\Http\Controllers;

use App\Models\Category;

class CategoryController extends Controller
{
    public function getPosts($id)
    {
        $category = Category::findOrFail($id);
        $posts = $category->posts;

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

```
<h2>Category: {{ $category->name }}</h2>
```

```
@if ($posts->isEmpty())
   <p>No posts found for this category.</p>
@else
   <ul>
      @foreach ($posts as $post)
         <li>{{ $post->title }}</li>
      @endforeach
   </ul>
@endif
```

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.

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
   // ...

   public function latestPost()
   {
      return $this->posts()->latest()->first();
   }

   // ...
}
```

```
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
   // ...

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

```
    }

    // ...
}


$category = Category::find($categoryId);

$latestPost = $category->latestPost();
```

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:

```
@foreach($categories as $category)
    <h2>{{ $category->name }}</h2>

    @if($category->posts->count() > 0)
        <div>
            <h3>{{ $category->posts->first()->title }}</h3>
            <p>{{ $category->posts->first()->content }}</p>
            <p>Published on: {{ $category->posts->first()->created_at }}</p>
        </div>
    @else
        <p>No posts found for this category.</p>
    @endif

    <hr>
@endforeach




}
```