Assignment_18

Task 1:

Artisan command: **php artisan make:migration create_categories_table**

Migration file:

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create( table: 'categories', function (Blueprint $table) {
            $table->id();
            $table->string( column: 'name',  length: 50);
            $table->timestamp( column: 'created_at')->useCurrent();
            $table->timestamp( column: 'updated_at')->useCurrent()->useCurrentOnUpdate();
        });
    }
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists( table: 'categories');
    }
};
```

Task 2:

Artisan command: **php artisan make:model Category**

Model file:

```php
class Category extends Model
{
    protected $table = 'categories'; // Specify the table name

    protected $fillable = ['name', 'description']; // Define the fillable properties

    // Define the relationships
    public function posts()
    {
        return $this->hasMany( related: Post::class); //suppose  u have a Post model
    }
}
```

Task 3:

Artisan Command: **php artisan make:migration add_category_id_to_posts_table --table=posts**

Migration file:

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table( table: 'posts', function (Blueprint $table) {
            $table->unsignedBigInteger( column: 'category_id')->after( column: 'id');

            $table->foreign( columns: 'category_id')
                ->references( columns: 'id')
                ->on( table: 'categories')
                ->cascadeOnUpdate()
                ->restrictOnDelete();
        });
    }

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

Task 4:

Post belongsTo category:

```php
class Post extends Model
{
    public function category()
    {
        return $this->belongsTo( related: Category::class);
    }
}
```

A category can have multiple posts:

```php
class Category extends Model
{
    protected $table = 'categories'; // Specify the table name

    protected $fillable = ['name', 'description']; // Define the fillable properties

    // Define the relationships
    public function posts()
    {
        return $this->hasMany( related: Post::class); //suppose  u have a Post model
    }

}
```

Task 5:

```php
class PostController extends Controller
{
    function postsByCategory(){
        $posts = Post::with( relations: 'category')->get();
    }

//    example
//    foreach ($posts as $post) {
//    echo $post->title;
//    echo $post->category->name;
//    }
}
```

Task 6:

Method in Post model to get the total number of posts belonging to a specific category. that should accept the category ID as a parameter and return the count:

```php
//    Task 6
    public function countPostsByCategory($categoryId)
    {
        return $this->where('category_id', $categoryId)->count();
    }
}
```

Task 7:

Route:

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

Use SoftDeletes on Post model:

```php
    use HasFactory, SoftDeletes;
```

Method in controller:

```php
//    Task 7
    public function delete($id)
    {
        $post = Post::findOrFail($id);
        $post->delete();

        return redirect()->route( route: 'posts.index')->with('success', 'Post deleted successfully.');
    }
```

Task 8:

Method for SoftDeletedPosts:

```php
//    Task 7
    public static function getSoftDeletedPosts()
    {
        return self::onlyTrashed()->get();
    }
```

Task 9:

Posts.blade.php file:

Category.php | PostController.php | Post.php | Posts.blade.php | web.php | App.blade.php

```blade
@extends('')


@section('content')

    @foreach ($posts as $post)
        <h2>{{ $post->title }}</h2>
        <p>{{ $post->content }}</p>
        <p>Category: {{ $post->category->name }}</p>
    @endforeach

@endsection
```

Task 10:

Route:

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

Controller:

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

        return view( view: 'category.posts', compact( varname: 'category', _: 'posts'));
    }
}
```

Task 11:

Method in Category Model:

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

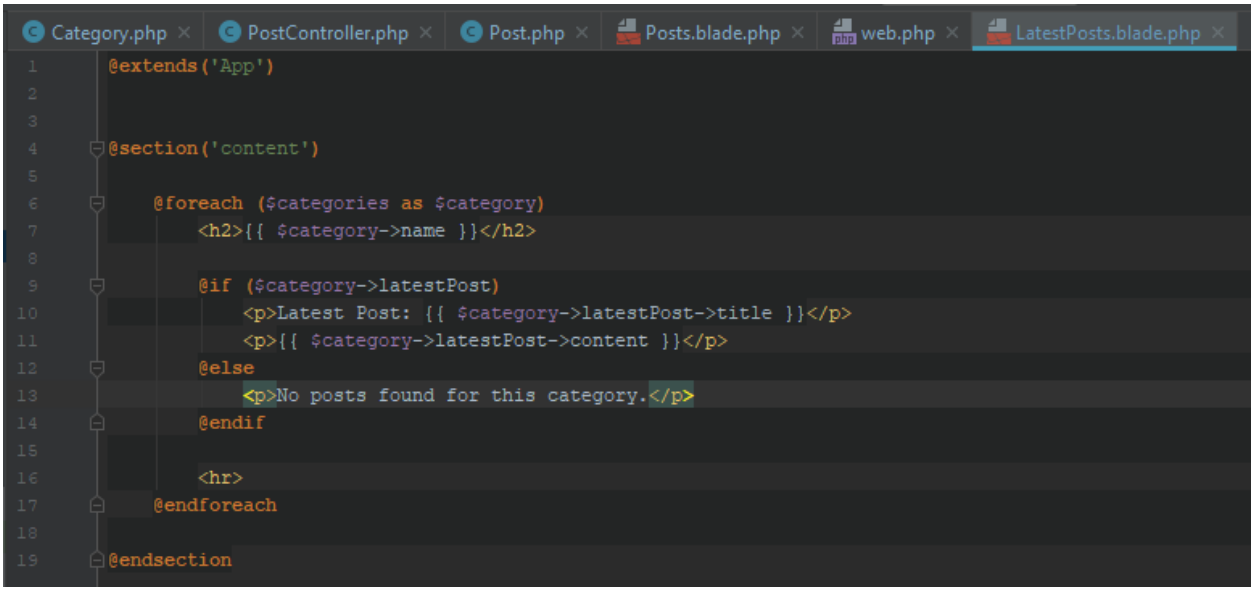
Route:

```php
Route::get( uri: '/catLatestPost/{id}', function (Request $request){
    $category = Category::find(1); // Replace 1 with the desired category ID
    $latestPost = $category->latestPost;

    if ($latestPost) {
        echo $latestPost->title;
        // Access other properties of the latest post as needed
    } else {
        echo "No posts found for this category.";
    }
});
```

Task 12:

Blade File:



Controller Method: