1.ManytoMany

2.ManytoManyMigration

At first we have to create a tags table

**php artisan make:migration create\_tags\_table**

**the**

**php artisan make:migration createBlogPostTagTable**

**Relation between tags and BlogPost Model**

**public function up()**

**{**

**Schema::create('blog\_post\_tag', function (Blueprint $table) {**

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

**$table->unsignedBigInteger('blog\_post\_id')->index();**

**$table->foreign('blog\_post\_id')**

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

**->on('blog\_posts')**

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

**$table->unsignedBigInteger('tag\_id')->index();**

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

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

**->on('tags')**

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

**$table->timestamps();**

**});**

**}**

**3.Defining ManytoMany on models**

**Go to BlogPost.php**

**public function tags(){**

**return $this->belongsToMany(Tag::class);**

**}**

**Go to Tag.php**

**public function blogPost(){**

**$this->belongsToMany(BlogPost::class);**

**}**

**4.Assosiating Model in ManytoMany**

**tinker code**

**$tag1=new App\Tag;**

**$tag1->name='Science';**

**$tag1->save();**

**$blogPost=BlogPost::find(8);**

**$blogPost->tags()->attach($tag1);**

**In case of attach it wont care it will attach same tags to different blogposts**

**in case of**

**syncwithoutDetaching it will checek if there is any duplication**

**If we want to detach all tags just simply write**

**$blogPost->tags()->detach();**

**5.Querying ManytoMany Relations and pivot tables**

**in BlogPost.php in tags function**

**public function tags(){**

**return $this->belongsToMany(Tag::class)->withTimestamps();**

**}**

**To access pivot tags you can write $tags[0]->pivot();**

**To access tags to specific blogposts**

**$tag=App\Tag::find(3);**

**$posts=$tag->blogPost;**

**Instead pivot if you want to use tagged propety you can use as tagged**

**in BlogPost.php**

public function tags(){

return $this->belongsToMany(Tag::class)->withTimestamps()->as('tagged');

}

public function blogPost(){

return $this->belongsToMany(BlogPost::class)->withTimestamps()->as('tagged');

}

$tag=App\Tag::find(3);

To find which app is related to BlogPost

$tag->BlogPost();

$post=BlogPost::find(8);

$post->tags();

**6.Practical using of Tags with Blade Component**

**At first in componment we will have to create**

tags.blade.php

**<p>**

**@foreach ($tags as $tag)**

**<a href="" class="badge badge-success">{{$tag->name}}</a>**

**@endforeach**

**</p>**

**In AppServiceProvider.php**

**public function boot(){**

**Blade::component('component.tags','tags');**

**}**

**in PostsController.php**

**in public function index(){**

**'posts' => BlogPost::latest()->withCount('comment')->with('user')->with('tags')->get()**

**}**

**in public function show(id){**

**$blogPost=Cache::remember("blog-post-{$id}", 60, function () use($id) {**

**return BlogPost::with('comment')->with('tags')->with('user')->findorFail($id);**

**});**

**}**

**7.List of BlogPosts by tag**

**We need to create a new Controller PostTagController.php**

**public function index(tagId){**

**}**

**Add a new route Web.php**

**Route::get('posts/tag/{id}','PostController@index')->name('posts.name.index);**

**in component/tags.blade.php**

**<p>**

**@foreach ($tags as $tag)**

**<a href="{{route('posts.tags.index',['tag'=>$tag->id]) }}" class="badge badge-success badge-lg">{{$tag->name}}</a>**

**@endforeach**

**</p>**

**in PostTagController.php**

**public function index($tag)**

**{**

**$tag= Tag::findorFail($tag);**

**return view('posts.index',**

**['posts'=>$tag->blogPost,**

**'most\_commented'=>[],**

**'mostActive'=>[],**

**'mostActiveLastMonth'=>[],**

**]**

**);**

**}**

**7.Blade View Composers**

**In http we will have to create a new composer folder and create a new file named ActivityComposer.php**

**In activityComposer.php we need to use**

**namespace App\Http\ViewComposers;**

**in ActivityComposer.php we need to use**

**class ActivityComposer{**

**public function compose(View $view) {**

**$most\_commented=Cache::remember('blog-post-commented', now()->addSeconds(30), function () {**

**return BlogPost::mostCommented()->take(5)->get();**

**});**

**$mostActive=Cache::remember('users-most-active', now()->addSeconds(30), function () {**

**return User::WithMostBlogPosts()->take(5)->get();**

**});**

**$mostActiveLastMonth=Cache::remember('users-most-active-last-month', now()->addSeconds(30), function () {**

**return User::withMostBlogPostsLastMonth()->take(5)->get();**

**});**

**$view->with('most\_commented',$most\_commented);**

**$view->with('mostActive',$mostActive);**

**$view->with('mostActiveLastMonth',$mostActiveLastMonth);**

**}**

**}**

**We can remove all this code from PostsController.php we will remove the most acommented most active inside index**

**and empty arrays in PostTagController.php**

**123.View Composer with @include**

**Indie posts add \_activity.blade.php**

**Take everyting inside col-md-4 of index.blade.php**

**and refactor put it into \_activity.blade.php**

**We also need to refactor show.blade.php**

**divide it with col-8 and col-4**

**in col-4 @include posts.\_activity**

**If you want to make it available to each and every blade in APPServiceProvider.php**

**view()->composer(['\*'],ActivityComposer::class);**

**8.ManytoManySeeding**

**make a new seeder php artisan make:seeder TagsTableSeeder**

**make another seeder to implement tag table seeder**

**php artisan make:seeder BlogPostTagTableSeeder**

**BlogPostTagTableSeeder.php**

**public function run()**

**{**

**//**

**$tagCount=Tag::all()->count();**

**if( 0 === $tagCount){**

**$this->command->info('no tags found,skip assigning tags to BlogPosts');**

**return;**

**}**

**$minimum=(int)$this->command->ask('Minimum ow many tags you want on this BlogPost?',0);**

**$maximum=min((int)$this->command->ask(' Maximum how many tags you want on this BlogPost?',$tagCount),$tagCount);**

**BlogPost::all()->each(function (BlogPost $post) use($minimum,$maximum){**

**$take=random\_int($minimum,$maximum);**

**$tags=Tag::inRandomOrder()->take($take)->get()->pluck('id');**

**$post->tags()->sync($tags);**

**});**

**}**