$php artisan make:migration create\_(TABLE-NAME)\_table

মানে হল এটা দিয়ে প্রজেক্ট এর ডাটাবেইজ ফোল্ডার এ টেবিল ফাইল তৈরি করে।

$php artisan migrate

মানে হল তৈরি করা টেবিল ফাইল ডাটাবেইজ এ আনা।

$ php artisan migrate:fresh

মানে হল যত গুলো টেবিল আছে সবগুলোর ডাটা ডিলিট করে সম্পূর্ণ নতুন করে দিবে।

$php artisan migrate:refresh

মানে হল তৈরি করা ডাটাবেইজ এর ফাইল গুলো আবার নতুন করে সেট হবে। টেবিলের সব ডাটা হারিয়ে যাবে এতে।

$php artisan migrate:reset

মানে হল ডাটা বেইজ থেকে সব টেবিল রিমুভ হবে যাবে মাইগ্রেশন টেবল বাদে।

$php artisan migrate:rollback

মানে হল লাস্ট মাইগ্রেট করে তৈরি করা টেবিল গুলো ডিলিট করে দিবে ডাটা সব। অন্য টেবিলের ডাটা হারাবে না।

$php artisan migrate:status

মানে হল কি কি টেবিল চালু আছে দেখাবে আর কোনটা কততম বার তৈরি করা হইছে তা দেখাবে ব্রাঞ্চ এর মাধ্যমে।

**Faker**

$php artisan make:migration create\_students\_table

প্রথমে ডাটাবেইজ এ একটা টেবল বানাব উপরের কমান্ট লাইনটা রান করে। এরপর migrations ফোল্ডার থেকে সেই ফাইলটা এডিট করে প্রয়োজনীয় কলাম অ্যাড করে নিচের কমান্ট টা রান করতে হবে।

$php artisan migrate

এখন ওই টেবল এ Fack ডাটা ঢুকাব Faker Class ইউজ করে। প্রথমে একটা মডেল ক্লাস তৈরি করা লাগবে তাই নিচের কমেন্ট টা ইউজ করব।

$php artisan make:model Students

Use Carbon\Carbon; এই লাইন টি model ক্লাসের উপরে অ্যাড করে দিব Carbon ক্লাস পাওয়ার জন্য। এরপর factory তৈরি করব নিচের লাইন টা রান করে।

$php artisan make:factory StudentsFactory

Use Carbon\Carbon; এই লাইন টি Factory ক্লাসে অ্যাড করে দিব Carbon ক্লাস পাওয়ার জন্য।

$factory->define(App\Student::class, function (Faker $faker) {

return [

];

}

এখানে define এ model ক্লাস টা দেখিয়ে দিতে হবে। এরপর return এ

'firstname' => $faker->firstName,

এইভাবে একটার পর একটা টেবিলের কলামের জন্য $faker এ নাম দেখে বসাতে হবে যেমনঃ firstName ।

এরপর Seeder তৈরি করব নিচের কমেন্ট দিয়ে।

$php artisan make:seeder StudentsTableSeeder

এরপর run() এ factory(App\Students::class, 5)->create(); লিখব আর model class টা দেখিয়ে দিব। এরপর কইটা ডাটা ঢুকাতে চাই সেই সংখ্যা নির্ধারণ করে দিব। এরপর Seeder Class টা DatabaseSeeder ফাইল এ দেখিয়ে দিতে হবে।

$this->call(StudentsTableSeeder::class);

সব কাজ শেষে autoload টা রান করে দিব সব কিছু ঠিক মত পাওয়ার জন্য।

$composer dump-autoload

সবশেষে নিচের কমান্ট টা রান করে ডাটাবেইজ এ ডাটা ঢুকাব।

$php artisan db:seed

এটা রান করলে DatabaseSeeder এ লিখা কোড টা রান হয়ে ডাটা ইন্সারট করবে। DatabaseSeeder এর লাইন টা না লিখেও StudentsTableSeeder ক্লাস টা রান করা যাবে তার জন্য---

$php artisan db:seed –class= StudentsTableSeeder

লিখতে হবে। এরপর ডাটা ইন্সারট হবে।

**// php artisan list //**

Available commands:

clear-compiled Remove the compiled class file

down Put the application into maintenance mode

dump-server Start the dump server to collect dump information.

env Display the current framework environment

help Displays help for a command

inspire Display an inspiring quote

list Lists commands

migrate Run the database migrations

optimize Cache the framework bootstrap files

preset Swap the front-end scaffolding for the application

serve Serve the application on the PHP development server

tinker Interact with your application

up Bring the application out of maintenance mode

app

app:name Set the application namespace

auth

auth:clear-resets Flush expired password reset tokens

cache

cache:clear Flush the application cache

cache:forget Remove an item from the cache

cache:table Create a migration for the cache database table

config

config:cache Create a cache file for faster configuration loading

config:clear Remove the configuration cache file

db

db:seed Seed the database with records

event

event:cache Discover and cache the application's events and listeners

event:clear Clear all cached events and listeners

event:generate Generate the missing events and listeners based on regist

ration

event:list List the application's events and listeners

key

key:generate Set the application key

make

make:auth Scaffold basic login and registration views and routes

make:channel Create a new channel class

make:command Create a new Artisan command

make:controller Create a new controller class

make:event Create a new event class

make:exception Create a new custom exception class

make:factory Create a new model factory

make:job Create a new job class

make:listener Create a new event listener class

make:mail Create a new email class

make:middleware Create a new middleware class

make:migration Create a new migration file

make:model Create a new Eloquent model class

make:notification Create a new notification class

make:observer Create a new observer class

make:policy Create a new policy class

make:provider Create a new service provider class

make:request Create a new form request class

make:resource Create a new resource

make:rule Create a new validation rule

make:seeder Create a new seeder class

make:test Create a new test class

migrate

migrate:fresh Drop all tables and re-run all migrations

migrate:install Create the migration repository

migrate:refresh Reset and re-run all migrations

migrate:reset Rollback all database migrations

migrate:rollback Rollback the last database migration

migrate:status Show the status of each migration

notifications

notifications:table Create a migration for the notifications table

optimize

optimize:clear Remove the cached bootstrap files

package

package:discover Rebuild the cached package manifest

queue

queue:failed List all of the failed queue jobs

queue:failed-table Create a migration for the failed queue jobs database tab

le

queue:flush Flush all of the failed queue jobs

queue:forget Delete a failed queue job

queue:listen Listen to a given queue

queue:restart Restart queue worker daemons after their current job

queue:retry Retry a failed queue job

queue:table Create a migration for the queue jobs database table

queue:work Start processing jobs on the queue as a daemon

route

route:cache Create a route cache file for faster route registration

route:clear Remove the route cache file

route:list List all registered routes

schedule

schedule:run Run the scheduled commands

session

session:table Create a migration for the session database table

storage

storage:link Create a symbolic link from "public/storage" to "storage/

app/public"

vendor

vendor:publish Publish any publishable assets from vendor packages

view

view:cache Compile all of the application's Blade templates

view:clear Clear all compiled view files

| **Command** | **Description** |
| --- | --- |
| $table->bigIncrements('id'); | Auto-incrementing UNSIGNED BIGINT (primary key) equivalent column. |
| $table->bigInteger('votes'); | BIGINT equivalent column. |
| $table->binary('data'); | BLOB equivalent column. |
| $table->boolean('confirmed'); | BOOLEAN equivalent column. |
| $table->char('name', 100); | CHAR equivalent column with an optional length. |
| $table->date('created\_at'); | DATE equivalent column. |
| $table->dateTime('created\_at'); | DATETIME equivalent column. |
| $table->dateTimeTz('created\_at'); | DATETIME (with timezone) equivalent column. |
| $table->decimal('amount', 8, 2); | DECIMAL equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->double('amount', 8, 2); | DOUBLE equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->enum('level', ['easy', 'hard']); | ENUM equivalent column. |
| $table->float('amount', 8, 2); | FLOAT equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->geometry('positions'); | GEOMETRY equivalent column. |
| $table->geometryCollection('positions'); | GEOMETRYCOLLECTION equivalent column. |
| $table->increments('id'); | Auto-incrementing UNSIGNED INTEGER (primary key) equivalent column. |
| $table->integer('votes'); | INTEGER equivalent column. |
| $table->ipAddress('visitor'); | IP address equivalent column. |
| $table->json('options'); | JSON equivalent column. |
| $table->jsonb('options'); | JSONB equivalent column. |
| $table->lineString('positions'); | LINESTRING equivalent column. |
| $table->longText('description'); | LONGTEXT equivalent column. |
| $table->macAddress('device'); | MAC address equivalent column. |
| $table->mediumIncrements('id'); | Auto-incrementing UNSIGNED MEDIUMINT (primary key) equivalent column. |
| $table->mediumInteger('votes'); | MEDIUMINT equivalent column. |
| $table->mediumText('description'); | MEDIUMTEXT equivalent column. |
| $table->morphs('taggable'); | Adds taggable\_id UNSIGNED INTEGER and taggable\_type VARCHAR equivalent columns. |
| $table->multiLineString('positions'); | MULTILINESTRING equivalent column. |
| $table->multiPoint('positions'); | MULTIPOINT equivalent column. |
| $table->multiPolygon('positions'); | MULTIPOLYGON equivalent column. |
| $table->nullableMorphs('taggable'); | Adds nullable versions of morphs() columns. |
| $table->nullableTimestamps(); | Alias of timestamps() method. |
| $table->point('position'); | POINT equivalent column. |
| $table->polygon('positions'); | POLYGON equivalent column. |
| $table->rememberToken(); | Adds a nullable remember\_token VARCHAR(100) equivalent column. |
| $table->smallIncrements('id'); | Auto-incrementing UNSIGNED SMALLINT (primary key) equivalent column. |
| $table->smallInteger('votes'); | SMALLINT equivalent column. |
| $table->softDeletes(); | Adds a nullable deleted\_at TIMESTAMP equivalent column for soft deletes. |
| $table->softDeletesTz(); | Adds a nullable deleted\_at TIMESTAMP (with timezone) equivalent column for soft deletes. |
| $table->string('name', 100); | VARCHAR equivalent column with a optional length. |
| $table->text('description'); | TEXT equivalent column. |
| $table->time('sunrise'); | TIME equivalent column. |
| $table->timeTz('sunrise'); | TIME (with timezone) equivalent column. |
| $table->timestamp('added\_on'); | TIMESTAMP equivalent column. |
| $table->timestampTz('added\_on'); | TIMESTAMP (with timezone) equivalent column. |
| $table->timestamps(); | Adds nullable created\_at and updated\_atTIMESTAMP equivalent columns. |
| $table->timestampsTz(); | Adds nullable created\_at and updated\_atTIMESTAMP (with timezone) equivalent columns. |
| $table->tinyIncrements('id'); | Auto-incrementing UNSIGNED TINYINT (primary key) equivalent column. |
| $table->tinyInteger('votes'); | TINYINT equivalent column. |
| $table->unsignedBigInteger('votes'); | UNSIGNED BIGINT equivalent column. |
| $table->unsignedDecimal('amount', 8, 2); | UNSIGNED DECIMAL equivalent column with a precision (total digits) and scale (decimal digits). |
| $table->unsignedInteger('votes'); | UNSIGNED INTEGER equivalent column. |
| $table->unsignedMediumInteger('votes'); | UNSIGNED MEDIUMINT equivalent column. |
| $table->unsignedSmallInteger('votes'); | UNSIGNED SMALLINT equivalent column. |
| $table->unsignedTinyInteger('votes'); | UNSIGNED TINYINT equivalent column. |
| $table->uuid('id'); | UUID equivalent column. |
| $table->year('birth\_year'); | YEAR equivalent column. |