Migration Assignment-16

1. To create a new Laravel project named "MigrationAssignment" using the Laravel command-line interface (CLI), you can follow these steps:

*Open a command line interface (e.g., Terminal, Command Prompt) on your machine.

*Navigate to the directory where you want to create the Laravel project.

*Run the following command to create a new Laravel project named "MigrationAssignment":

```
composer create-project --prefer-dist laravel/laravel MigrationAssignment
```

2. To create a new migration file named "create_products_table" responsible for creating a table called "products" with the specified columns, you can follow these steps:

Make sure you have the Laravel project set up and you are in the project's root directory in the command line interface.

Run the following artisan command to create a new migration file:

```
php artisan make:migration create_products_table --create=products
```

This command will generate a new migration file with the specified name and the --create=products option indicates that the migration is responsible for creating the "products" table.

Open the newly created migration file located in the database/migrations directory. The file should have a name similar to 2023_05_22_123456_create_products_table.php, with a timestamp as a prefix.

Inside the up() method of the migration file, define the table structure using the Schema Builder's fluent API. Your migration file should look like this:

```
use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;


class CreateProductsTable extends Migration

{

  public function up()

  {
      Schema::create('products', function (Blueprint $table) {
          $table->id();
          $table->string('name');
          $table->decimal('price', 8, 2);
          $table->text('description');
          $table->timestamps();
      });
  }

  public function down()
  {
      Schema::dropIfExists('products');
  }
}
```

In the code snippet above, the up() method is responsible for creating the "products" table. The id() method creates an auto-incrementing primary key column. The string() method is used to define the "name" column as a string. The decimal() method creates a decimal column for the "price" field, specifying the total digits (8) and the number of decimal places (2). The text() method is used to define the "description" column as a text field. The timestamps() method adds the created_at and updated_at columns for tracking the creation and update dates.

The down() method is used to define the table rollback operation in case you need to undo the migration.

Save the migration file.

Now you can run the migration to create the "products" table by executing the following artisan command:

```
php artisan migrate
```

After running the migration, the "products" table will be created in your database with the specified columns. Make sure to adjust the migration file and table structure according to your specific needs.

3. Make sure you are in the root directory of your Laravel project in the command line interface.

Run the following command to execute the migration:

```
php artisan migrate
```

This command will run all pending migrations in the database/migrations directory.

Laravel will execute the migration and create the "products" table in the database based on the migration file you created. The output in the command line will display the status of the migration.

After running the migration successfully, the "products" table with the specified columns will be created in your database.

Make sure your database configuration is properly set up in the .env file located in the root directory of your Laravel project. Laravel will use the database connection details specified in the .env file to create the table.

4. Open the existing migration file "create_products_table" located in the database/migrations directory.

Inside the up() method of the migration file, use the addColumn() method of the Schema Builder to add the new "quantity" column to the "products" table. Update the migration file as follows:

```
use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;


class CreateProductsTable extends Migration

{

  public function up()

  {

    Schema::create('products', function (Blueprint $table) {

      $table->id();

      $table->string('name');

      $table->decimal('price', 8, 2);

      $table->text('description');

      $table->integer('quantity')->nullable();

      $table->timestamps();

    });

  }


  public function down()
```

```
    {
        Schema::dropIfExists('products');
    }
}
```

In the updated migration file, the addColumn() method is used to add the new "quantity" column to the "products" table. The integer('quantity')->nullable() defines the "quantity" column as an integer that allows null values.

Save the modified migration file.

5. Make sure you are in the root directory of your Laravel project in the command line interface.

Run the following artisan command to create a new migration file:

```
php artisan make:migration add_category_to_products_table --table=products
```

This command will generate a new migration file with the specified name and the --table=products option indicates that the migration is responsible for modifying the "products" table.

Open the newly created migration file located in the database/migrations directory. The file should have a name similar to 2023_05_22_123456_add_category_to_products_table.php, with a timestamp as a prefix.

Inside the up() method of the migration file, use the addColumn() method of the Schema Builder to add the new "category" column to the "products" table. Update the migration file as follows:

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddCategoryToProductsTable extends Migration
{
    public function up()
    {
        Schema::table('products', function (Blueprint $table) {
            $table->string('category', 50)->after('description');
        });
    }


    public function down()
    {
        Schema::table('products', function (Blueprint $table) {
            $table->dropColumn('category');
        });
    }
```

}In the updated migration file, the addColumn() method is used inside the up() method to add the new "category" column to the "products" table. The string('category', 50) defines the "category" column as a string with a maximum length of 50 characters. The after('description') specifies that the new column should be added after the "description" column.

The down() method is used to define the rollback operation to drop the "category" column if needed.

Save the migration file.

```
php artisan migrate
```

After running the migration, the "products" table will be updated with the new "category" column. The column will be a string column with a maximum length of 50 characters.

6. Make sure you are in the root directory of your Laravel project in the command line interface.

Run the following command to execute the migration:

```
php artisan migrate
```

This command will run all pending migrations in the database/migrations directory.

Laravel will execute the migration and add the "category" column to the "products" table based on the migration file you created. The output in the command line will display the status of the migration.

After running the migration successfully, the "category" column will be added to the "products" table in your database.

Make sure your database configuration is properly set up in the .env file located in the root directory of your Laravel project. Laravel will use the database connection details specified in the .env file to run the migration and modify the table.

If you have already run migrations before creating this new migration file, Laravel will only run the new migration and make the necessary changes to the "products" table.

7. Make sure you are in the root directory of your Laravel project in the command line interface.

Run the following artisan command to create a new migration file:

```
php artisan make:migration create_orders_table
```

This command will generate a new migration file with a name similar to 2023_05_22_123456_create_orders_table.php, with a timestamp as a prefix.

Open the newly created migration file located in the database/migrations directory.

Inside the up() method of the migration file, use the create() method of the Schema Builder to create the "orders" table with the specified columns. Update the migration file as follows:

```
use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;


class CreateOrdersTable extends Migration

{

    public function up()

    {

        Schema::create('orders', function (Blueprint $table) {

            $table->id();

            $table->unsignedBigInteger('product_id');

            $table->integer('quantity');

            $table->timestamps();


            // Foreign key constraint

            $table->foreign('product_id')->references('id')->on('products');

        });

    }
    public function down()
    {
        Schema::dropIfExists('orders');
    }
}
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateOrdersTable extends Migration
{
    public function up()
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('product_id');
            $table->integer('quantity');
            $table->timestamps();

            // Foreign key constraint
            $table->foreign('product_id')->references('id')->on('products');
        });
    }

    public function down()
    {
        Schema::dropIfExists('orders');
    }
}
```
In the updated migration file, the create() method is used to create the "orders" table. The id() method is used to create the auto-incrementing primary key column. The unsignedBigInteger('product_id') creates the foreign key column for the product relationship. The integer('quantity') creates the column to store the quantity of products ordered. Finally, the timestamps() method is used to create the created_at and updated_at columns.

The foreign() method is used to define a foreign key constraint on the product_id column, referencing the id column of the products table.

The down() method is used to define the rollback operation to drop the "orders" table if needed.

8. Save the migration file.

```
php artisan migrate
```