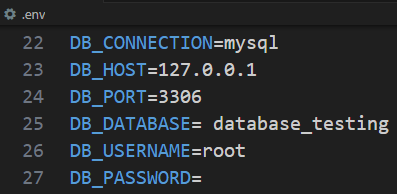
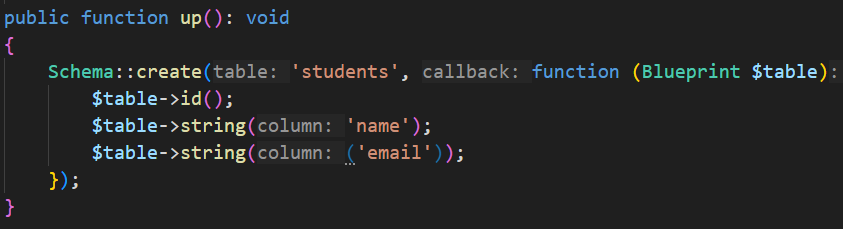
* Connect database:



* For creating a table



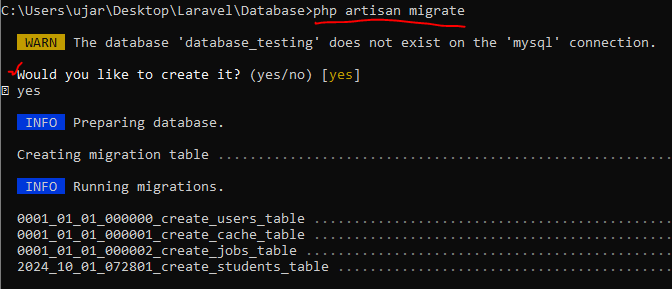


Now

If we write



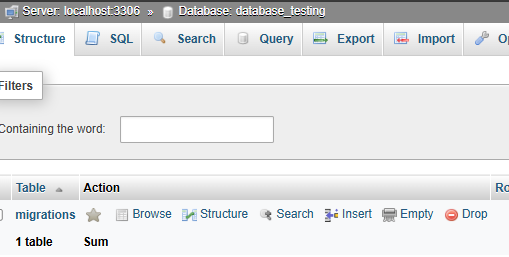
And doesn’t have a database, then it first create database,



If we want to remove last migration then,

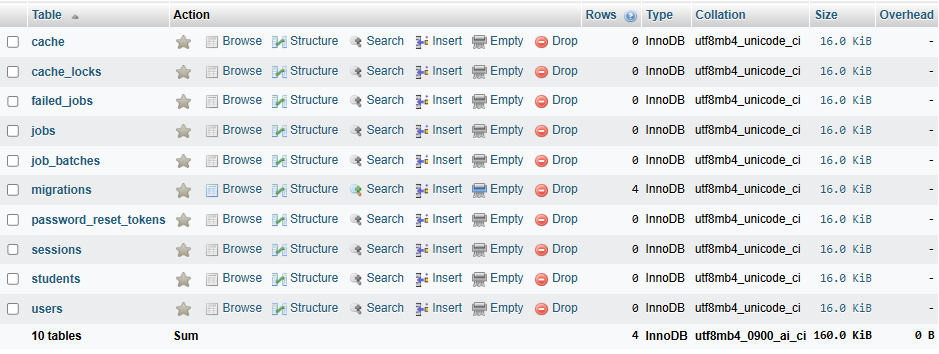


Then database remove previous created tables



If we again migrate then,





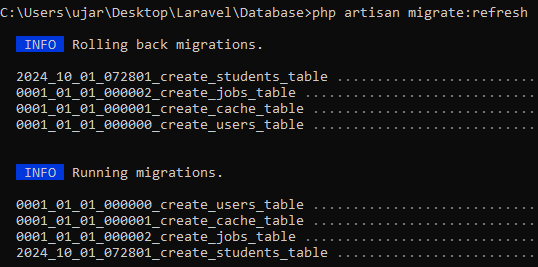
For reset all the table from database,



If we want to remove all the table from database and then again create tables to the database

Using one command





This is used for testing purpose

Same other command is

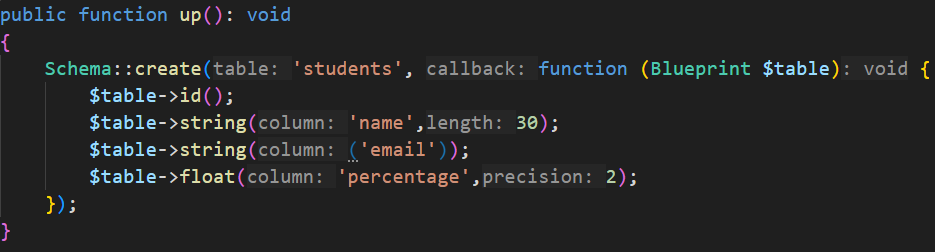


There are two types of modification on database

1. Column modification
2. Add new column
3. Rename column
4. Delete column
5. Change column order
6. Change datatype or size of column
7. Table modification
8. Rename Table
9. Delete table

Now we will add new column in our students table,

Previous migration file is,



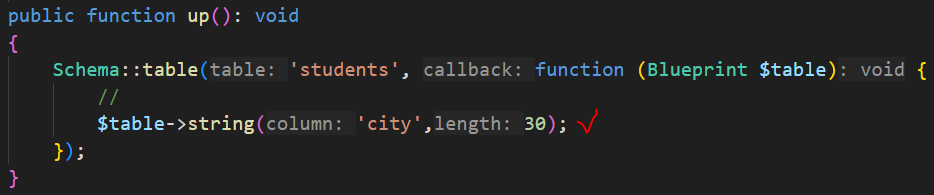
Now we add ‘city’ column in this table,

For this



Here “**--table=students**:” This option specifies that the migration will modify the **existing “students” table**.

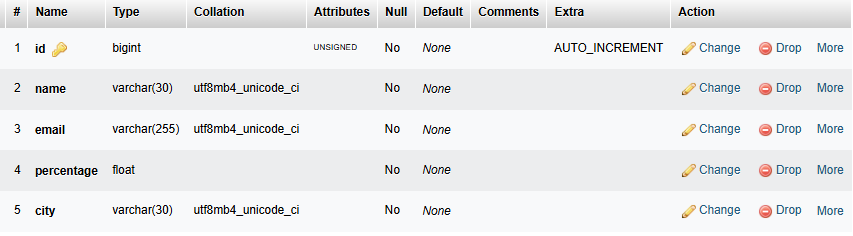
Now in update migration file,



Now we need to run migrate



Now

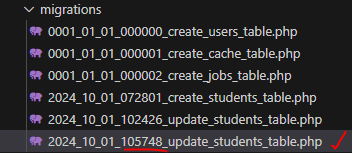


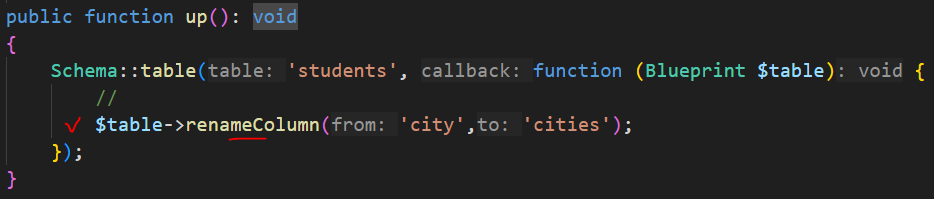
Now we need to change ‘column name’ for example we need to change ‘city’ column to ‘cities’

For this we again need to run the same command,



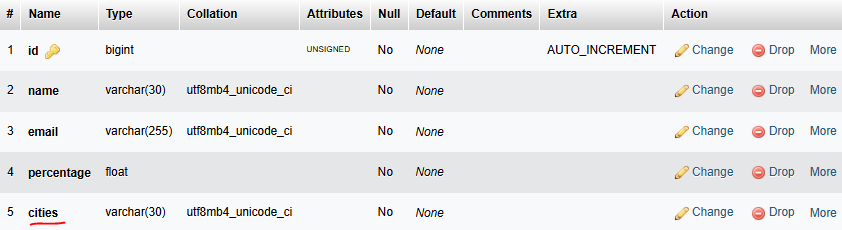
It will generate same name migration tables but now time is different





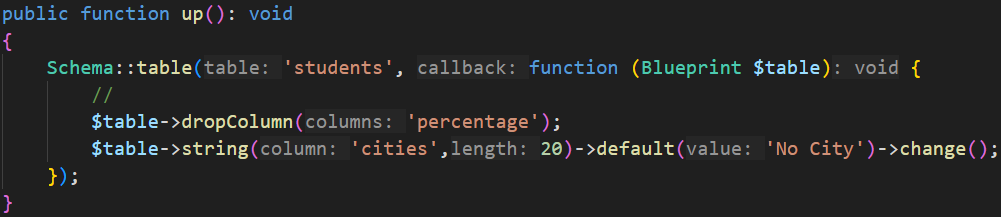
Now,



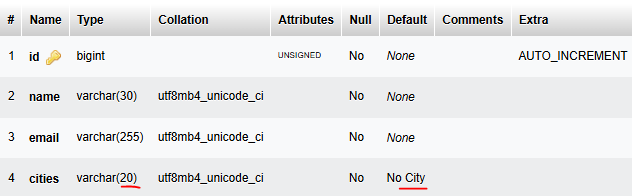


Now we need to drop ‘percentage’ column and modify ‘cities’ column we set default value here and change varchar size is 20 for this,

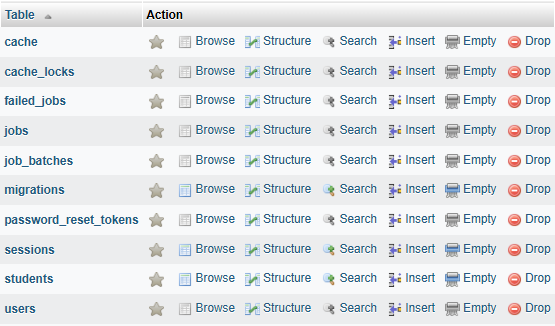






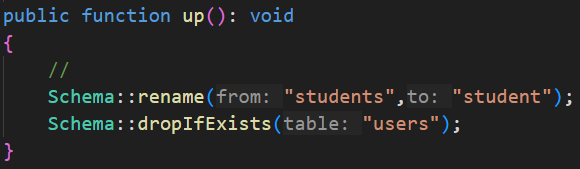


Now we rename a table name and drop a table,

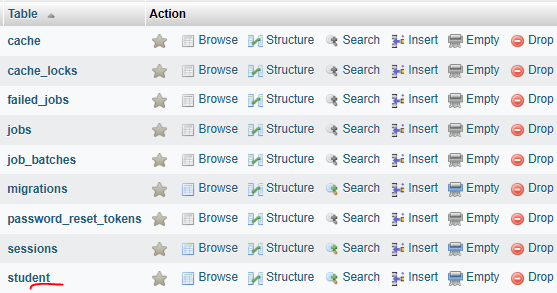


For this first we create a new migration file,

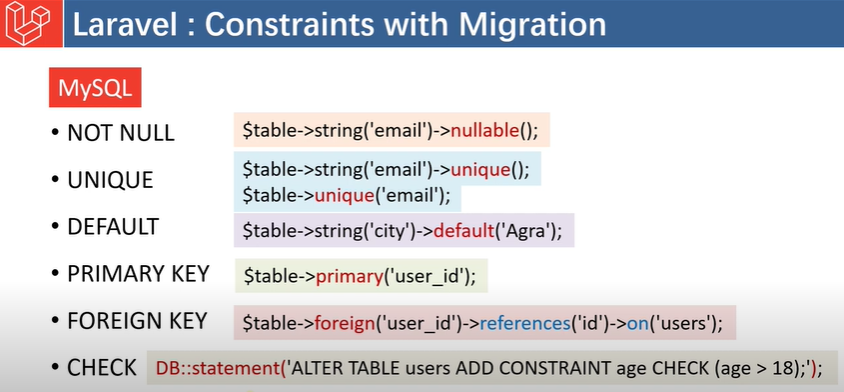






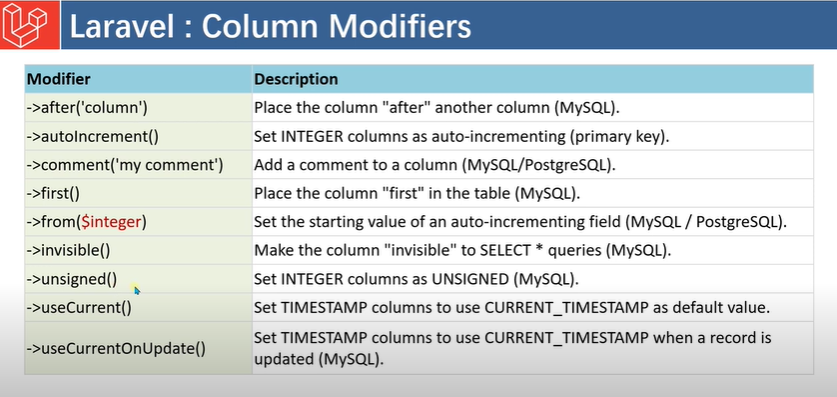


Constraints with Migration



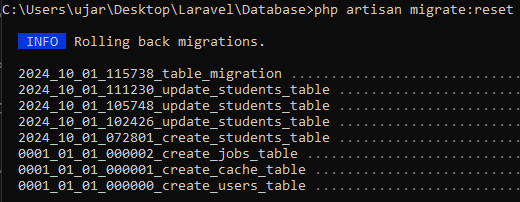
MySQL ->

1. Not Null (by default) if we want to make it null then ->nullable()
2. Unique
3. Default
4. Primary key
5. Foreign key
6. Check

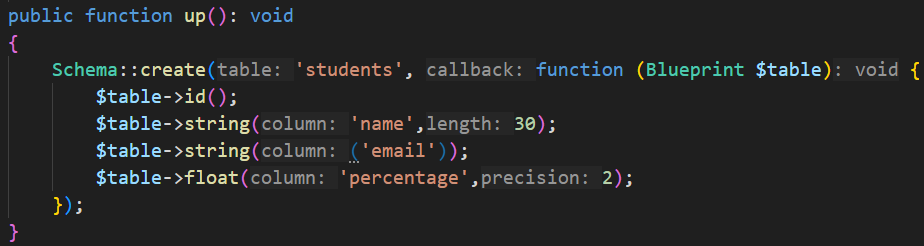


Now we see the example of constraints

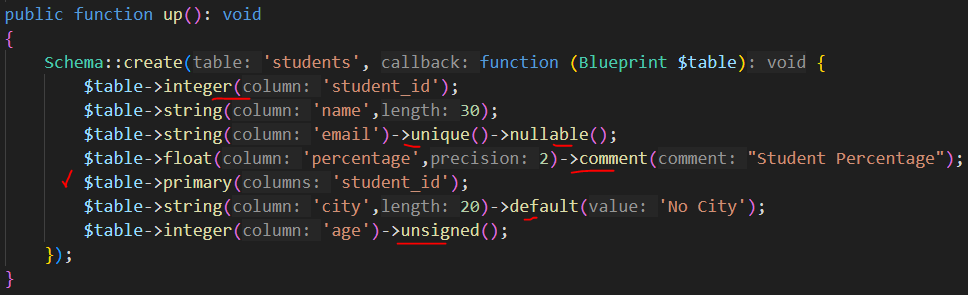
First we reset all the migrations from the database then we modify our students table



Previous create migration file

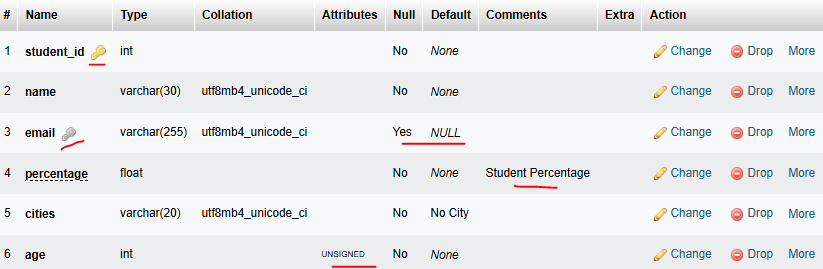


Updated file,



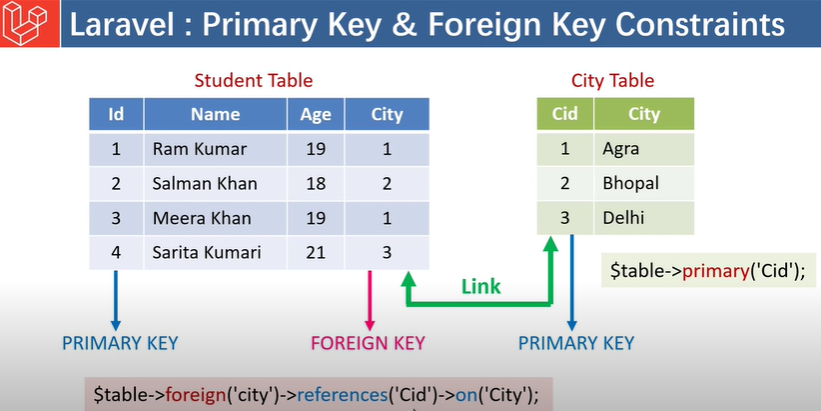
Here “->unsigned()” means the 'age' column should only store positive integers (no negative values allowed)





* Primary key and Foreign key Constraints

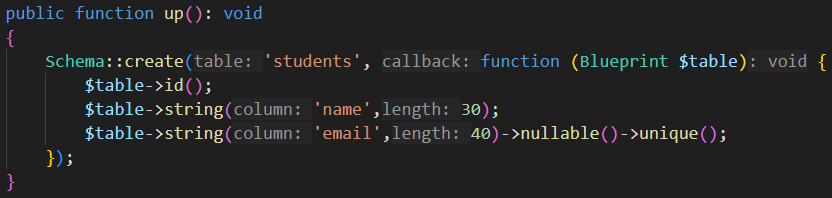
A table can have only one primary key and a foreign is always a primary key of another table.



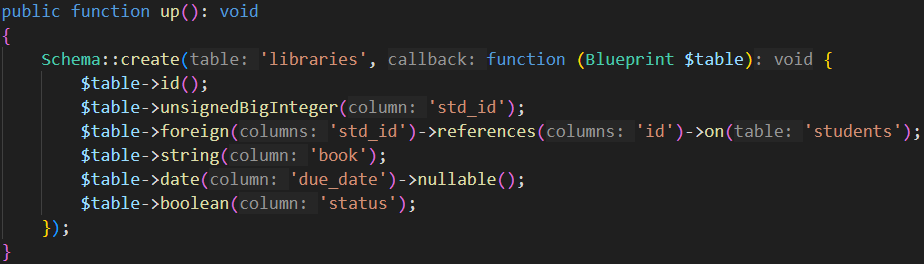
When we define foreign key, we set ‘references’ to ‘another tables column (which is primary key of that table)’ and set ‘on’ to ‘that table name’

Now we create two migrations files first







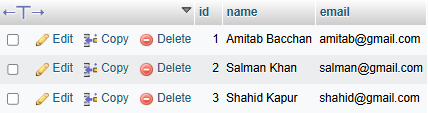


“unsignedBigInteger('std\_id')”: Adds a new column named ‘std\_id’ to the table. This column will store large integer values (big integers) that cannot be negative (unsigned).

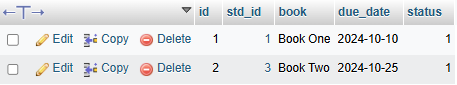
“date('due\_date')”: Adds a new column named “due\_date” to the table, which will store date values.

After migrate,

Students table

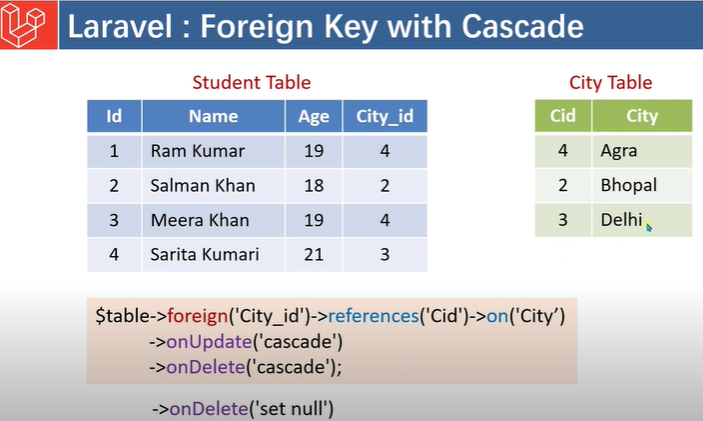


Libraries table



Now student id 1 & 3 are connected with other table, we can’t modify those rows it shows error but if we want to change student id 2 we can change it easily because this row is not connected with others table.

But if we want to change any value of a table row which is connected with another table and when we change the value of the table and we need to auto change others connected table we need to use “casecase”

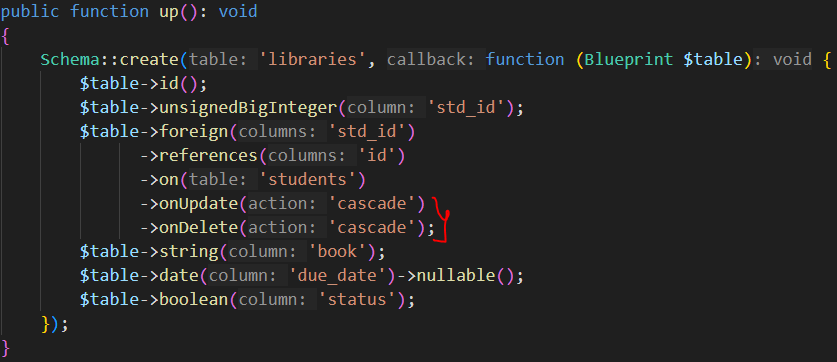


If we set onDelete(‘set null’) then deleted main table then connected tables row shows null.

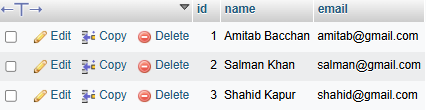
Now we see example,

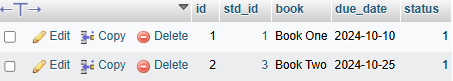


Now in libraries migration file,

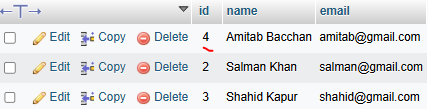


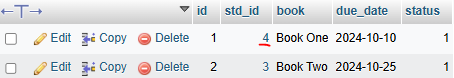
After migrate,





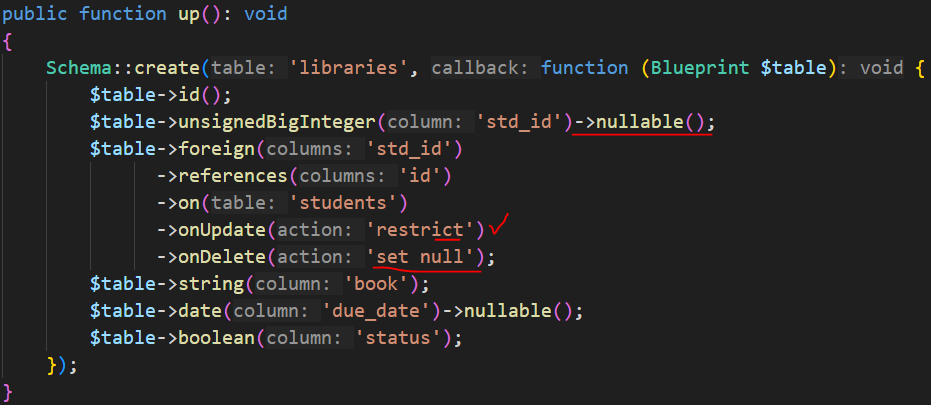
Now if we change student table id then,





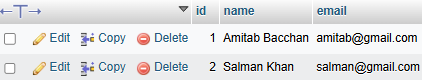
It’s changed

Now we want to use ‘restrict’ on update and ‘set null’ on delete

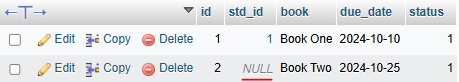


If we want to “set null” then the column must be “nullable()”

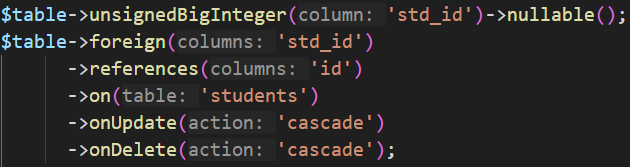
Now we can’t modify but we can delete



We delete id->3



Shorts-cut



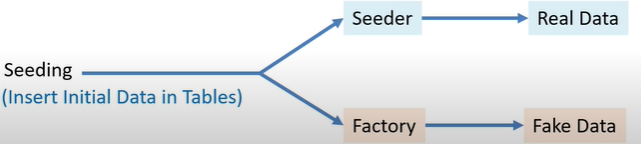
To



Here we use “student\_id” because our table name is “students”

We need to set the name related our primary keys table name otherwise not work it

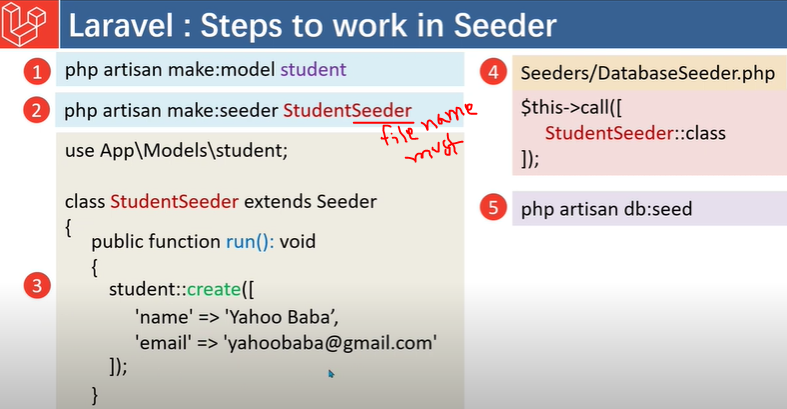
* Seeding: Insert initial data in tables. There are two way to seeding data



When we create database migrations file we can use name like “create\_students\_table” I mean we can use ‘s’ but when we create it’s model, we don’t need to use ‘s’

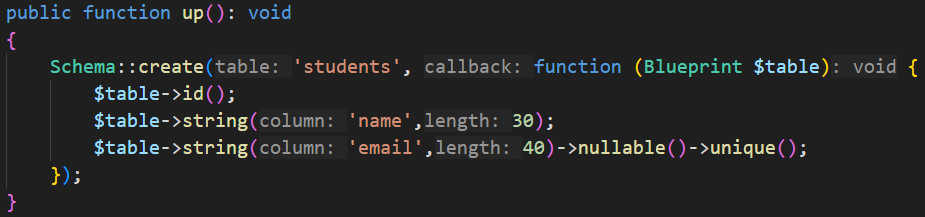
For example our model name is “student”

<- note-> ‘void’ function doesn’t have return type

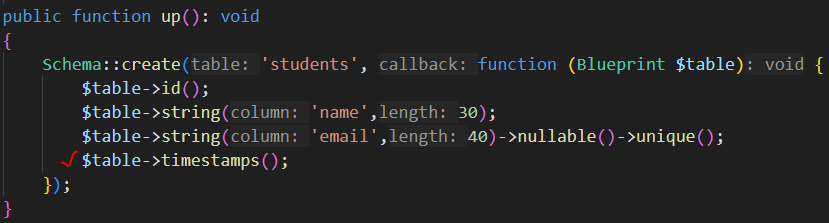


Now we see an example,

Our previous database table,



We change it a bit

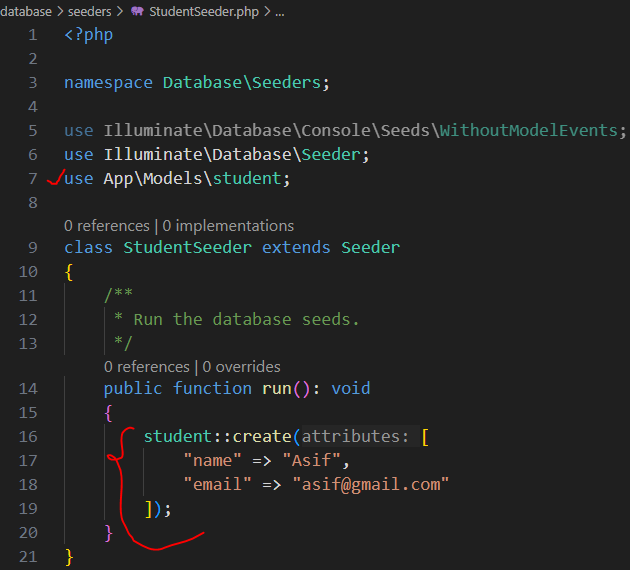


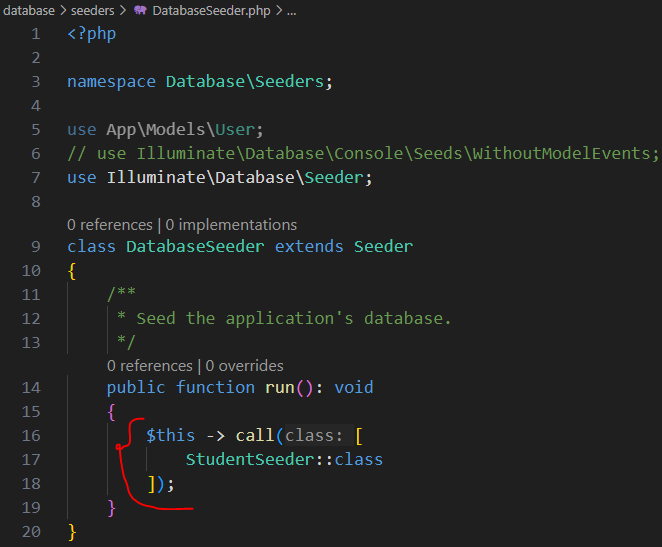
File name -> 

Now







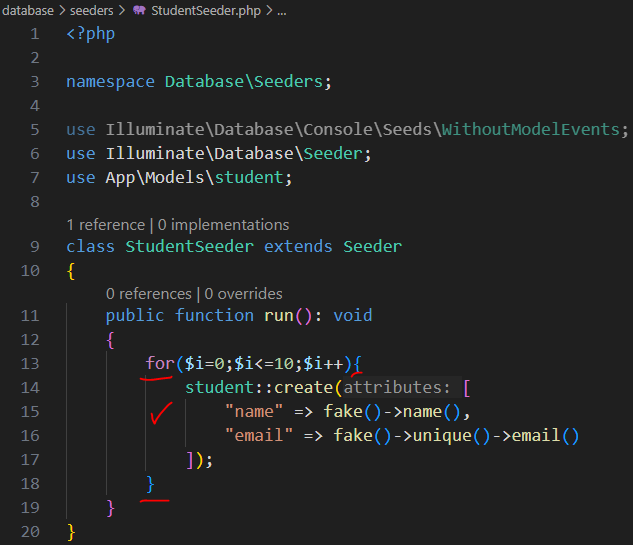
  
Now we need to write a command



After that

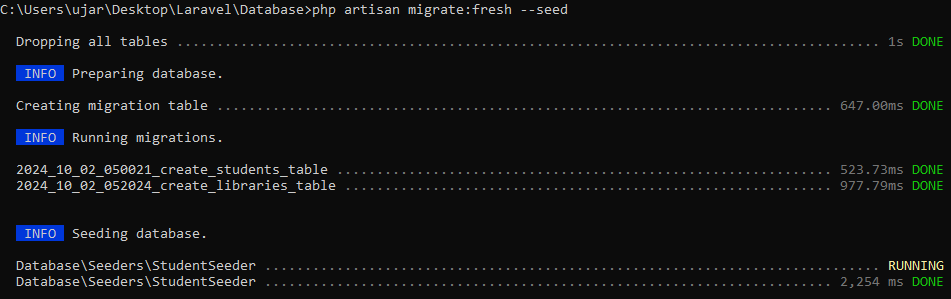


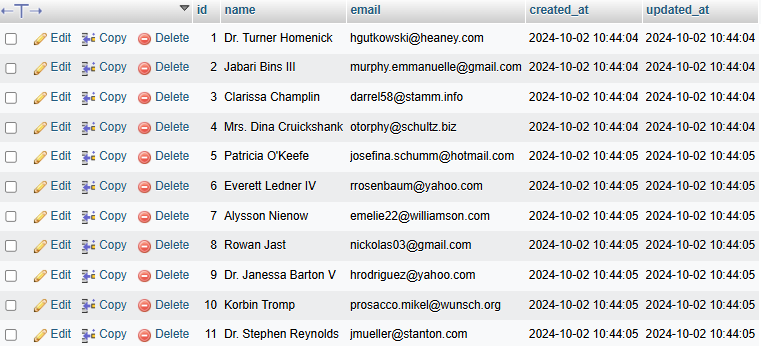
Now we add some fake data to our database for this, we need to open “StudentSeeder” file



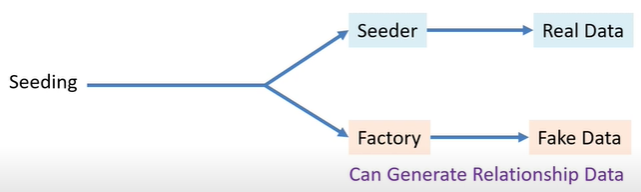
Now we need to write

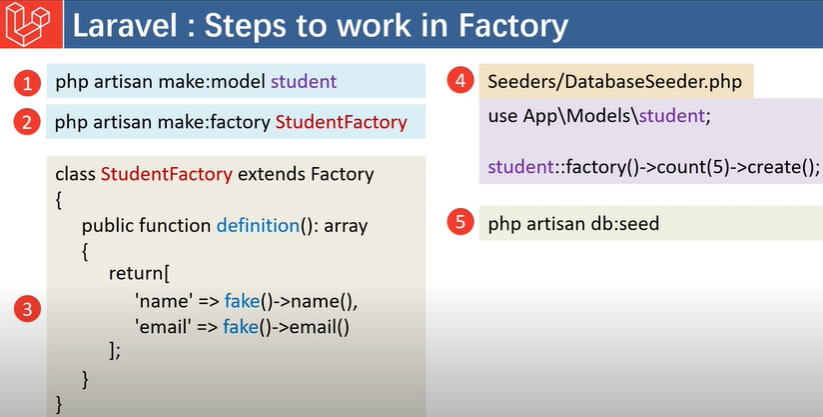






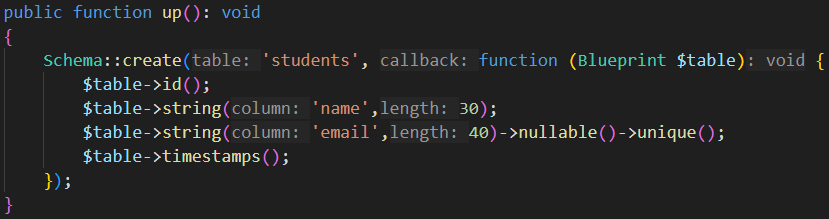
* Factory



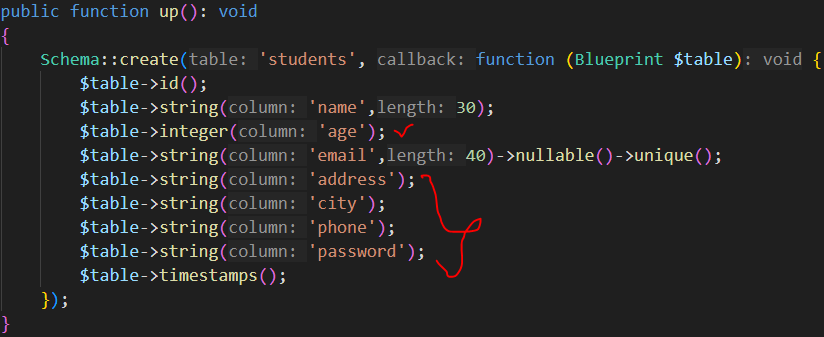


Now we are working with factory

Our previous “students” migration file is



We want to add some more column



We already have “student” model so we will create it

For factory,

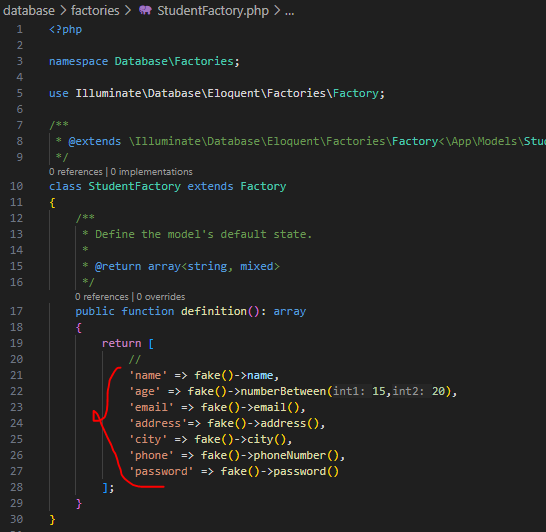


Now in StudentFactory,

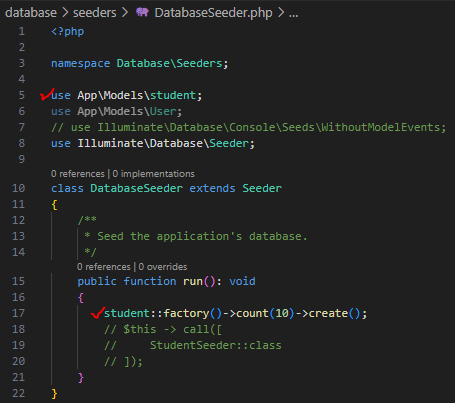
We need to add fake()->something

This something we will find in “fakephp.org” site

Now in Studentfactory

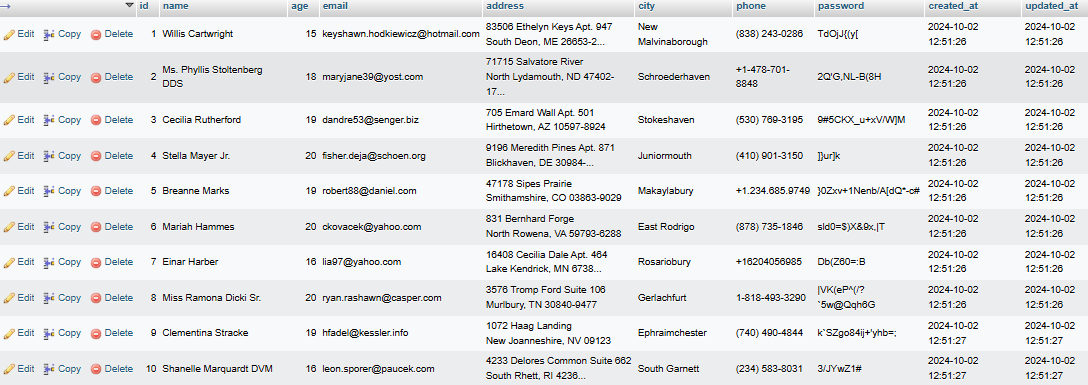


Now in “DataSeeder” file



Now,





18:32

Course: Laravel Model Factory Tutorial in Hindi / Urdu