**Sequelize Command:**

**Seqeulize CLI:**

npm install –save-deb sequelize-cli

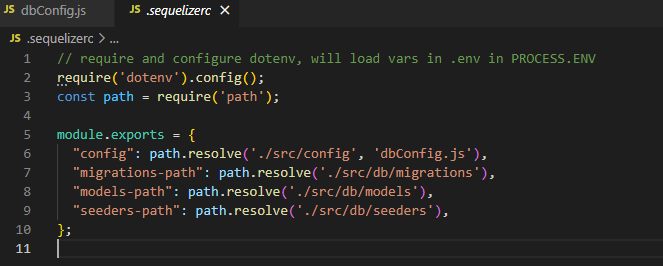
**How to generate migrations, models folder in project:**

npm install --save sequelize

npx sequelize-cli init

**New file at root directory:**

For create migration, models, or seeders we first need to create a file at root directory where we define our **paths** mean where migration or models will be created when we run command for create them.



**Migration:**

npx sequelize-cli migration:generate --name migration-name

**Running Migration: (**How to run migration**);**

npx sequelize-cli db:migrate

**Create Alter Migration:**

How to add column in existing table using migration?

In Up section:

await queryInterface.addColumn('patients', 'pharmaciesId',{

      type:Sequelize.UUID,

      references:{

        model:'pharmacies',

        key:'id'

      }

    });

In Down Section:

await queryInterface.removeColumn('patients', 'pharmaciesId')

**Seeders:**

If we want to give table to anyone then they can add it’s own entry.

Like:

 {

        id: '956bb347-63c2-420f-94d3-e8e55e6d149e',

        name: 'New South Wales',

        to: '2914',

        from: '0200',

        createdAt: new Date(),

        updatedAt: new Date(),

        code: 'NSW',

      },

Command:

npx sequelize-cli seed:generate –name seader-name

Run All the seeder.

npx sequelize-cli db:seed:all

For run specific seeder.

npx sequelize-cli db:seed –seed seed-name

Note:

When we run npx sequelize-cli db:seed:all this command multiple times then it put same data multiple time in our db.

**For undo Seeder:**

In undo case **down** function is run.

down: async (queryInterface, Sequelize) => { // eslint-disable-line no-unused-

     \*/

    await queryInterface.bulkDelete('States', null, {});

  },

npx sequelize-cli db:seed:undo:all

**Query Interface:**

Is used for create table, add column, change column, delete column, delete database in table etc.

await queryInterface.addColumn('patients', 'pharmacyId',{

      type:Sequelize.UUID,

      references:{

        model:'pharmacies',

        key:'id'

      }

    });

**Hooks:**

Hooks (also known as lifecycle events), are **functions which are called before and after calls in sequelize are executed**. For example, if you want to always set a value on a model before saving it, you can add a before Update, before Create, after Create, After Update etc hook.

**Where we use hook:**

We use hooks in **model** as third parameter. We can use hooks in two ways.

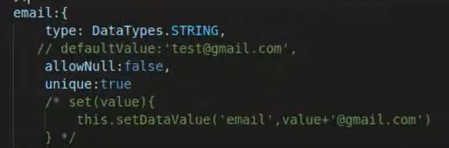


**Validation and Constraints:**

**Constraints:**

Condition at SQL level mean email entry must be not null or any other validation at sql level;

Define in model



**Validation:**

Sequelize level validation. Mean name send in db should be alphabetically like use in JavaScript.