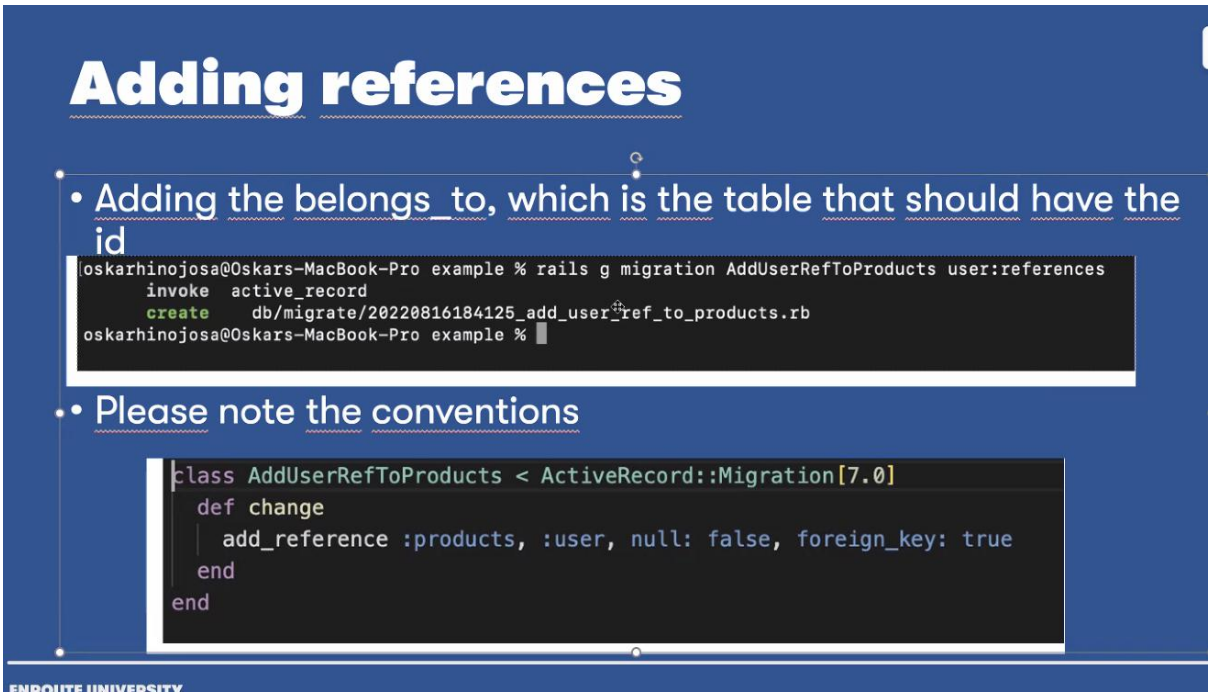
VESION -> you can say the timestamp, so you only do a specific migration.



1. Table on which we want to add the foreign key
2. the referenced table
3. foreign_key: true -> add the constraint

index: false -> really bad performance so try to avoid (sometimes will be necessary).

Join Tables: an explanation

- A product may have multiple customers and a customer may have multiple products

```
class CreateJoinTableCustomerProduct < ActiveRecord::Migration[7.0]
  def change
    create_join_table :customers, :products do |t|
      t.index [:customer_id, :product_id]
      t.index [:product_id, :customer_id]
    end
  end
end
```

They are Intermediary tables on many to many relationships.

A word of knowledge

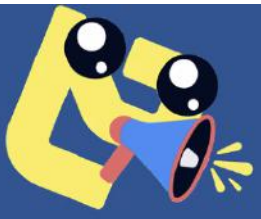
- Using rails g migration would mean you would have to create your own models
- To avoid this, simply use the model generator
  - The model generator will create the relevant table and model, it is an aid for helping scaffolding (creating multiple files at once)
  - Please know it would also create fixtures (tests)

scaffolding -> generate multiple with one command.
Creating using generators depends on the team, since it can create things that you don't actually will use.

How to specify your own index?

```
lass CreateUsers < ActiveRecord::Migration[7.0]
 def change
   create_table :users do |t|
     t.string :name, index: true
     t.string :email, index: { unique: true, name: 'unique_emails' }

     # in the last command, please note that you are also specifying the table
     # column name
   end
 end
nd
```

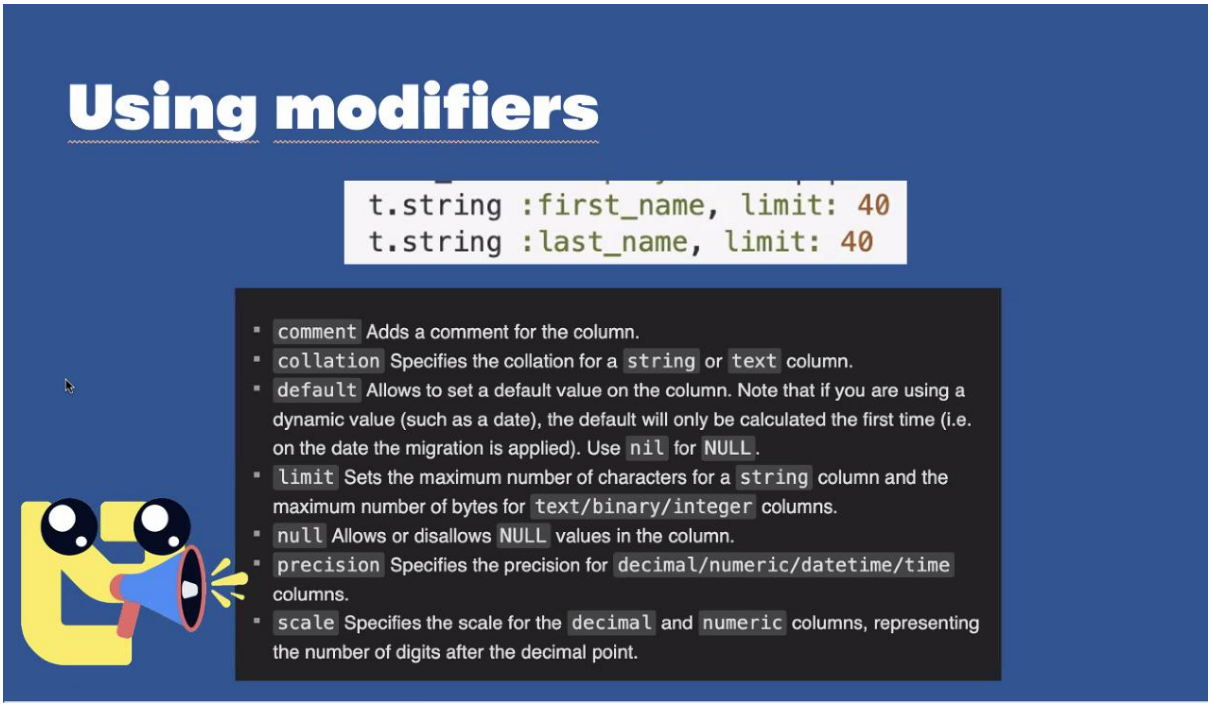
Modifying tables



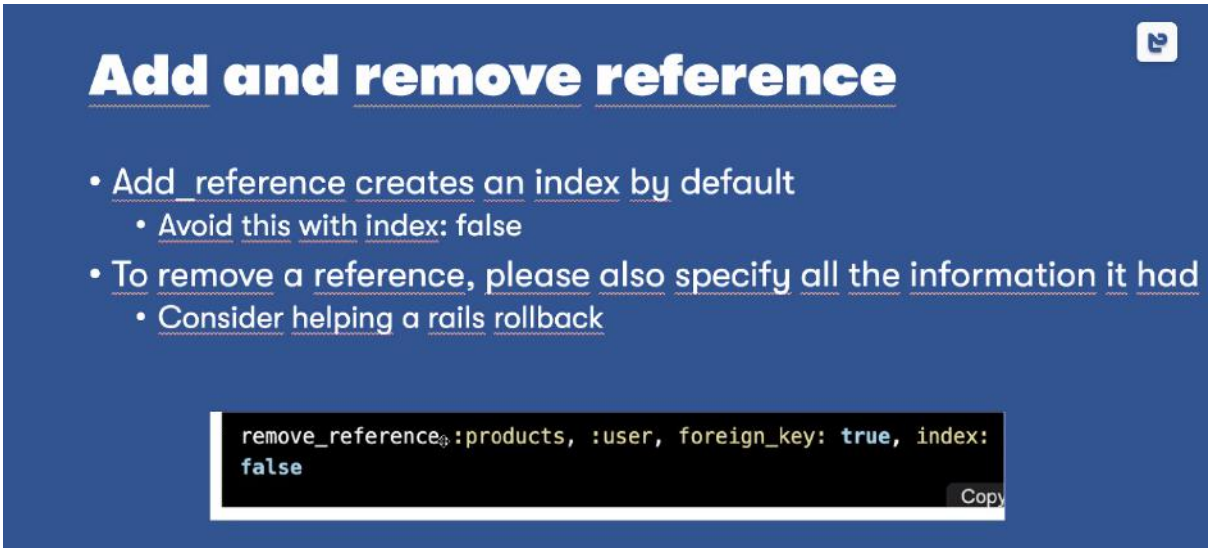
- Note that in the image, two columns were removed

```
change_table :products do |t|
  t.remove :description, :name
  t.string :part_number
  t.index :part_number
  t.rename :upccode, :upc_code
end
```

All these modifying commands are not necessary for us to memorize them.
You can have multiple actions on one line using a comma.



Default and limit are the most used ones.
Precision and scale -> make it loose information on rollback.



Reference is not the same as foreign key. The reference key is the column that the foreign key from one table is using to connect to the other table.

Adding foreign keys

- Just like in the last command, add_foreign_key first mentions the table in which the change will occur
  - In this case, articles is referencing to authors

```
add_foreign_key :articles, :authors
```

```
add_foreign_key :articles, :authors, column: :reviewer,
primary_key: :email
```

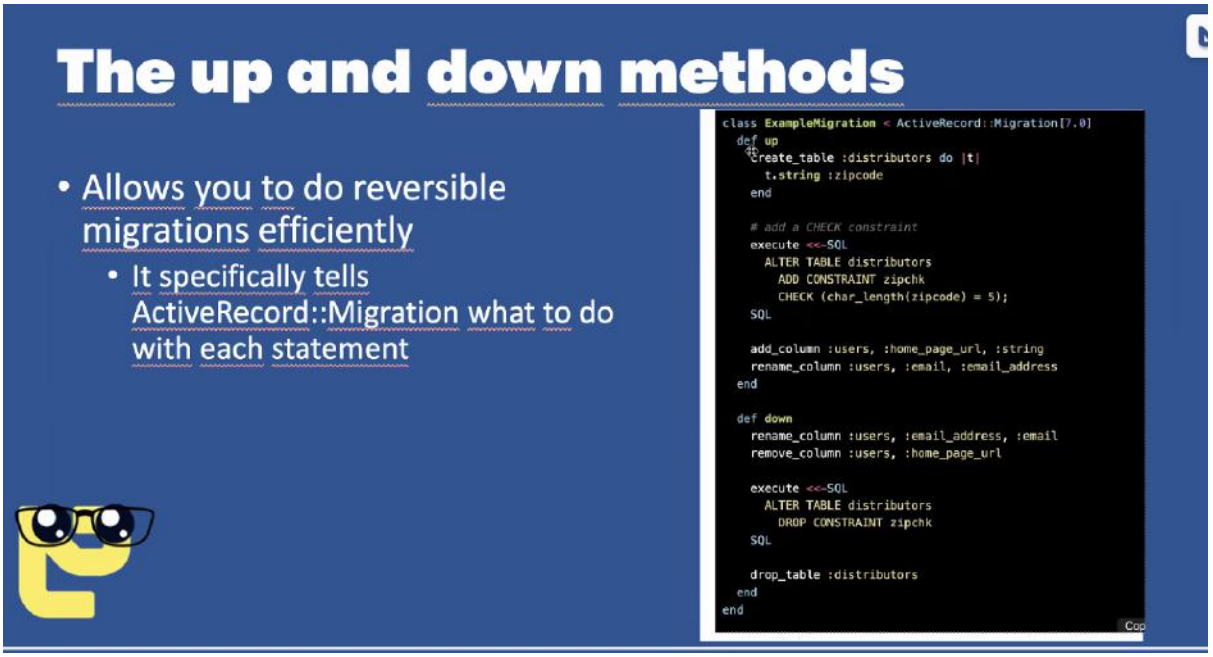
Removing foreign keys

```
# let Active Record figure out the column name
remove_foreign_key :accounts, :branches

# remove foreign key for a specific column
remove_foreign_key :accounts, column: :owner_id
```

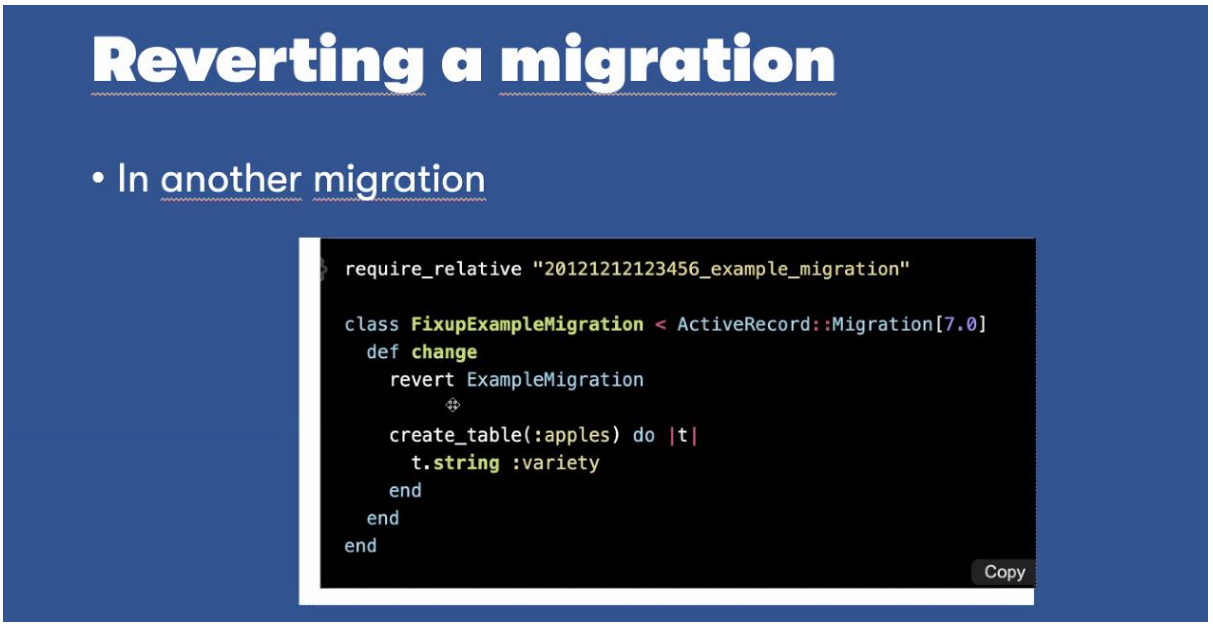
Executing a database statement

```
Product.connection.execute("UPDATE products SET price =
'free' WHERE 1=1")
```

Not a good practice at least it is stated by enterprise rules.



It is not a reversible migration since we used an sql statement.



Another way to do a reverse, using a relative, if you want to do a reverse on another migration you can use that code.
It depends if it is a bad practice to do a rollback, sometimes it is easier to rollback than creating a whole new migration.

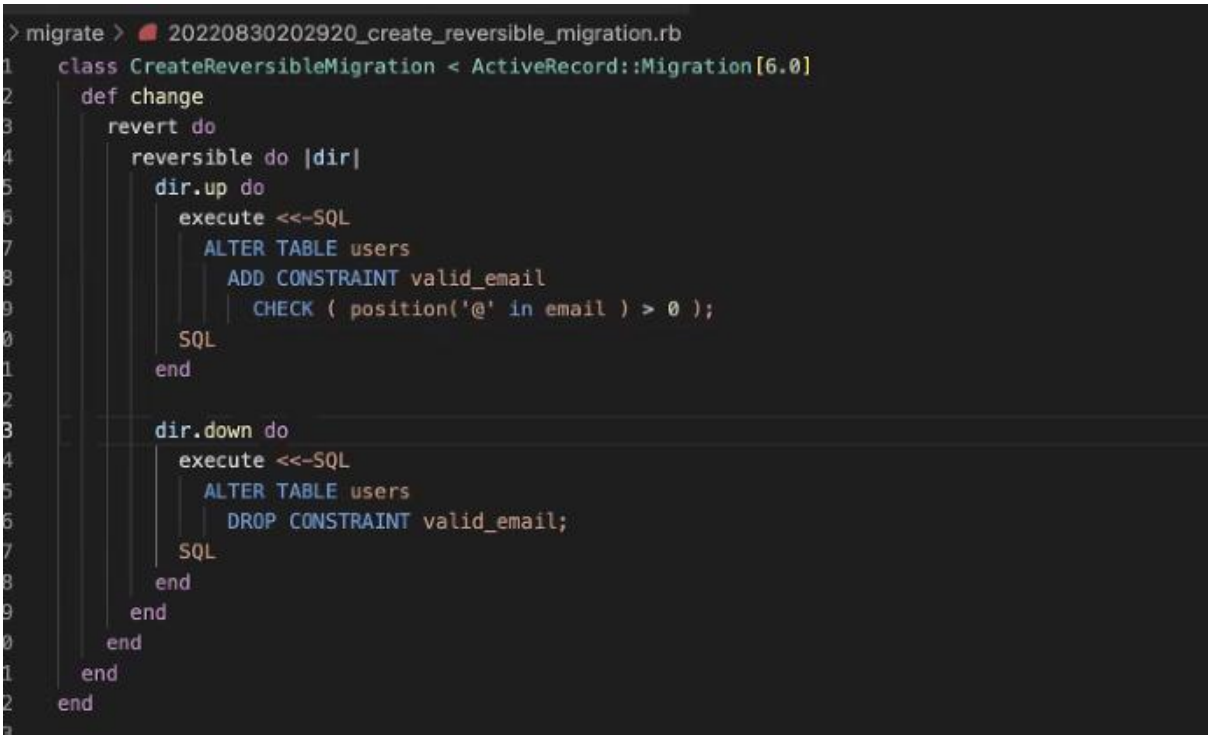
Using explicit SQL statements

```
class DontUseConstraintForZipcodeValidationMigration <
ActiveRecord::Migration[7.0]
  def change
    revert do
      # copy-pasted code from ExampleMigration
      reversible do |dir|
        dir.up do
          # add a CHECK constraint
          execute <<-SQL
            ALTER TABLE distributors
              ADD CONSTRAINT zipchk
                CHECK (char_length(zipcode) = 5);
          SQL
        end
        dir.down do
          execute <<-SQL
            ALTER TABLE distributors
              DROP CONSTRAINT zipchk
          SQL
        end
      end

      # The rest of the migration was ok
    end
  end
end
```

Using explicit SQL statements


```
class DontUseConstraintForZipcodeValidationMigration <
ActiveRecord::Migration[7.0]
  def change
    revert do
      # copy-pasted code from ExampleMigration
      reversible do |dir|
        dir.up do
          # add a CHECK constraint
          execute <<-SQL
            ALTER TABLE distributors
              ADD CONSTRAINT zipchk
                CHECK (char_length(zipcode) = 5);
          SQL
        end
        dir.down do
          execute <<-SQL
            ALTER TABLE distributors
              DROP CONSTRAINT zipchk
          SQL
        end
      end

      # The rest of the migration was ok
    end
  end
end
```


With explicit sql statements always try to use up and down.


migrate > 20220830202920_create_reversible_migration.rb
class CreateReversibleMigration < ActiveRecord::Migration[6.0]
def change
revert do
reversible do |dir|
dir.up do
execute <<-SQL
ALTER TABLE users
ADD CONSTRAINT valid_email
CHECK (position('@' in email) > 0);
SQL
end
dir.down do
execute <<-SQL
ALTER TABLE users
DROP CONSTRAINT valid_email;
SQL
end
end
end
end
end

db > migrate > 20220830202920_create_reversible_migration.rb

```ruby
class CreateReversibleMigration < ActiveRecord::Migration[6.0]
  def change
    revert do
      reversible do |dir|
        dir.up do
          execute <<-SQL
            ALTER TABLE users
              ADD CONSTRAINT valid_email
                CHECK ( position('@' in email ) > 0 );
          SQL
        end

        dir.down do
          execute <<-SQL
            ALTER TABLE users
              DROP CONSTRAINT valid_email;
          SQL
        end
      end
    end
  end


  def change

    revert do
      reversible do |dir|
        dir.up do
        end

        dir.down do
        end
      end
    end

  end


  def up
    execute <<-SQL
      ALTER TABLE users
        ADD CONSTRAINT valid_email
          CHECK ( position('@' in email ) > 0 );
    SQL
  end


  def down
    execute <<-SQL
      ALTER TABLE users
        DROP CONSTRAINT valid_email;
    SQL
  end
```

```
oskarhinojosa@Oskars-MacBook-Pro ecommerce % rails db:rollback STEP=2
warning ../package.json: No license field
```

Step -> it is the number of migrations to revert.

Notes on the CLI

- You can run specific migrations by using the timestamp at the end of the created file
  - Rails db:migrate VERSION=202208160000
- You can rollback the last migration or the last three migrations (using three, please note that three can be changed to anything)
  - Rails db:rollback [STEP=#]
- You can rollback and migrate again in a simple step
  - Rails db:migrate:redo [STEP=#]

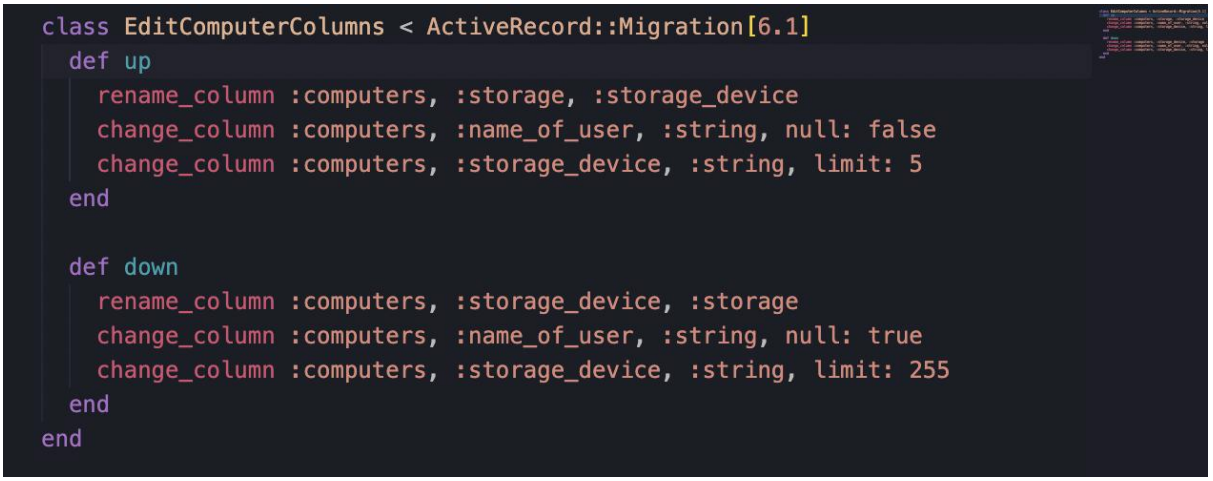
Irreversible action

> migrate > 20220831165015_edit_computer_columns.rb

```ruby
class EditComputerColumns < ActiveRecord::Migration[6.1]
  def change
    rename_column :computers, :storage, :storage_device
    change_column :computers, :name_of_user, :string, null: false
    change_column :computers, :storage_device, :string, limit: 5
  end
end
```

```
Caused by:
ActiveRecord::IrreversibleMigration:

This migration uses change_column, which is not automatically reversible.
To make the migration reversible you can either:
1. Define #up and #down methods in place of the #change method.
2. Use the #reversible method to define reversible behavior.


/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration/command_recorder.rb:100:in `inverse_of'
/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration/command_recorder.rb:80:in `record'
/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration/command_recorder.rb:112:in `change_column'
/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration.rb:929:in `block in method_missing'
/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration.rb:897:in `block in say_with_time'
/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration.rb:897:in `say_with_time'
/Users/luisroberto/.rvm/gems/ruby-3.0.4/gems/activerecord-6.1.6.1/lib/active_record/migration.rb:918:in `method_missing'
/Users/luisroberto/Documents/ENROUTE/RoR-Workshop/rails-exercises/migration-exercise/db/migrate/20220831165015_edit_computer_columns.rb:4:in `change'
```

What should have been done



```ruby
class EditComputerColumns < ActiveRecord::Migration[6.1]
  def up
    rename_column :computers, :storage, :storage_device
    change_column :computers, :name_of_user, :string, null: false
    change_column :computers, :storage_device, :string, limit: 5
  end

  def down
    rename_column :computers, :storage_device, :storage
    change_column :computers, :name_of_user, :string, null: true
    change_column :computers, :storage_device, :string, limit: 255
  end
end
```

URLs

https://millarian.com/rails/precision-and-scale-for-ruby-on-rails-migrations/
https://www.gapintelligence.com/blog/up-and-down-a-rails-migration/
https://medium.com/@kevinkarma55/rails-active-record-migration-change-method-vs-up-down-method-eaba011de3e9