Zadanie1. Stwórz pacjenta name i surname , w którym będzie możliwe dodawanie i wyświetlanie pacjentów.
Stwórz widoki ręcznie za pomocą form_for oraz w routes użyj resources.

Rails new zad1
rails g model Patient name:string surname:string

rails g controller patients show new create

resources :patients, only: [:show,:new,:create]

NEW.HTML.ERB:

```
<%= form_for(@patient) do |f| %>
  <div class="field">
    <%= f.label :name %><br>
    <%= f.text_field :name %>
  </div>
  <div class="field">
    <%= f.label :surname %><br>
    <%= f.text_field :surname %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
<% end %>
```

KONTROLER:

```
class PatientsController < ApplicationController
  def show
        @patient = Patient.find(params[:id])
  end

  def new
         @patient = Patient.new
  end

  def create
        @patient = Patient.new(patient_params)
        if @patient.save
                redirect_to @patient, notice: 'Patient was successfully created.'
        else
                render action: 'new'
```

```
        end
  end

  private

  def patient_params
     params.require(:patient).permit(:name, :surname)
  end

end
```

Zadanie2. Stwórz scaffold tworzący pracownika Employee , name , surname salary, commision.

Stwórz widok wyświetlający tylko pracowników , którzy zarabiają więcej niż 20000.

Użyj resources i collection do wyświelania pracowników zarabiających najlepiej.

Dodaj odpowiedni widok full_salary , który wyświetla całkowite wynagrodzenia pracownika (użyj tutaj member).

```
resources :employees do
    collection do
      get "highest_salary"
    end
    member do
      get "full_salary"
    end

  end

@employees = Employee.where("salary>=:salary",salary: 10000)
    render 'index'

<%= render 'show' %>

<h1> Full Salary: <%= @employee.full_salary %> </h1>.
```

Zadanie3. Zrefaktoryzuj kontroler, tak aby była metoda highest_salary na modelu.

Zadanie4. Zrefaktoryzuj kod, tak aby korzystał z scopes.

scope :highest_salary, -> { where("salary>=:salary",salary: 10000) }

Zadanie5. Stwórz scaffold Book oraz scaffold Author.

Stwórz widok , w którym będzie możliwe wybranie authora do danej książki.

Zadanie6. Stwórz ankietę z pytaniami i odpowiedzami. W jednym widoku powinno być możliwe dodawanie nowej ankiety wraz z pytaniami i odpowiedzami.

http://railscasts.com/episodes/196-nested-model-form-part-1

rails new zad6

rails g scaffold Survey name:string

rails g model question survey_id:integer content:text

rails g model answer question_id:integer content:string

```
class Survey < ActiveRecord::Base
  has_many :questions, :dependent => :destroy
  accepts_nested_attributes_for :questions
end

class Question < ActiveRecord::Base
  belongs_to :survey
  has_many :answers, :dependent => :destroy
  accepts_nested_attributes_for :answers
end
class Answer < ActiveRecord::Base
          belongs_to :question
end
```

```
def new
   @survey = Survey.new
    3.times do
      question = @survey.questions.build
```

```
      4.times { question.answers.build }
    end
  end

def survey_params
     params.require(:survey).permit(:name,questions_attributes:[:id,
:content,:answers_attributes =>[:question_id,:content]])
   end
```

FORM:

```
 <div class="field">
   <%= f.label :name %><br>
   <%= f.text_field :name %>
 </div>
 <%= f.fields_for :questions do |builder| %>
 <p>
   <%= builder.label :content, "Question" %><br />
   <%= builder.text_area :content, :rows => 3 %><br />
   <%= builder.fields_for :answers do |f| %>
     <p>
       <%= f.label :content, "Answer" %>
       <%= f.text_field :content %>
```

```
    </p>
   <% end %>
  </p>
  <% end %>
```

SHOW:

```
<p id="notice"><%= notice %></p>

<p>
  <strong>Name:</strong>
  <%= @survey.name %>
</p>

<% @survey.questions.each do |question| %>
        <li>
        <%=h question.content %>
        <ul>
        <% question.answers.each do |answer| %>
        <li><%=h answer.content %> </li>
        <% end %>
        </ul>


        </li>

<% end %>
```

Zadanie7. Stwórz scaffold z Użytkownikiem. Użytkownik powinen posiadać username , active (walidujemy string active bądź inactive).

Stwórz 2 scopy:
-active wyświetlający tylko użytkowników aktywnych
-inactive wyświetlający użytkowników nieaktywnych

scope **:active**, -> { where state: *'active'* }
scope **:inactive**, -> { where state: *'inactive'* }

Stwórz 2 widoki wyświetlający aktywnych użytkowników oraz widok wyświetlający nieaktywnych użytkowników.

Zadanie8. Stwórz scaffold z Mem (name:string description:text) oraz dodatkowo image z paperclipa , stwórz dodatkowy widok wyświetlający tylko memy w formacie medium 300 na 300 (użyj styles w Paperclip).