```
#loading packages
begin
    using PlutoUI    ,ExtendableGrids    ,VoronoiFVM    ,GridVisualize    , PlutoVista

    using HypertextLiteral
    using DifferentialEquations
    GridVisualize.default_plotter!(PlutoVista)
end;
```

0

```
begin
    #defining constants
    const k  = 100;
    const g = [0 -1];
    const c = 0.001;
    const α = 0.01;
    const λ = 0.01;
    const ρ_ref = 1;
    const T_ref = 0;
    const P_ref = 0;
end
```

Porous_medium (generic function with 2 methods)

```
function Porous_medium(N_x,N_y,T_heat,steady_state,tend = 10)

        #defining grid
        X = collect(range(0,300,length=N_x))
        Y = collect(range(0,150,length=N_y))
        grid=simplexgrid(X,Y)

        #defining flux
        function flux!(f,u, edge)
            f[1] = -ρ_ref*k*(u[1,1]-u[1,2]) -α*0.5*(u[2,1]+u[2,2])*project(edge,g)
            f[2] = λ*(u[2,1] - u[2,2])
        end

        #defining storage(what is under the timederivative)
        function storage!(f,u,node)
            f[1]=0
            f[2]=c*u[2]
        end

        #defining boundary conditions
        function bcondition!(f,u,bnode)
            boundary_dirichlet!(f,u,bnode,species=2,region=1,value=T_heat)
            boundary_dirichlet!(f,u,bnode,species=2,region=3,value=0)
            boundary_dirichlet!(f,u,bnode,species=1,region=3,value=0)
        end

        #defining voronoi system
        system=VoronoiFVM.System(grid;
        flux=flux!,
        bcondition = bcondition!,
        storage = storage!,
        species=[1,2])

        if steady_state
            #solving steadystate solution
            sol=VoronoiFVM.solve(system)
            nf=nodeflux(system,sol)
            return grid,sol,nf
        else
            #solving transient solution
            sol=VoronoiFVM.solve(system,inival = 0,times=
(0,tend),Δu_opt=T_heat,Δt_min=1.0e-2, Δt=1.0e-1)
            #inival=unknowns(system,inival=0)
            #problem = ODEProblem(system,inival,(0,tend))
            #odesol = DifferentialEquations.solve(problem)
            #sol=reshape(odesol,system)
            nf=nodeflux(system,sol[1])
            return grid,sol,nf
        end
end
```

```
(ExtendableGrids.ExtendableGrid{Float64, Int32};        , t: 21-element Vector{Float64}
 dim: 2 nodes: 2312 cells: 4422 bfaces: 200, edges: 6733    0.0
                                                            0.1
                                                            0.2
                                                            0.32
                                                            0.464
                                                            0.6368
                                                            0.84416
                                                            ⋮
                                                            5.94966
                                                            6.94966
                                                            7.94966
                                                            8.94966
                                                            9.94966
                                                           10.0
                                                          u: 21-element Vector{Matrix{F
                                                           [0.0 0.0 … 0.0 0.0; 0.0 0.0
                                                           [0.002483 0.002483 … -4.4982
                                                           [0.00268437 0.00268437 … -2.
                                                           [0.00291444 0.00291444 … -2.
                                                           [0.00317572 0.00317572 … -2.
                                                           [0.00347063 0.00347063 … -5.
                                                           [0.00380149 0.00380149 … -3.
                                                           ⋮
                                                           [0.00879543 0.00879543 … -5.
                                                           [0.009481 0.009481 … -4.5189
                                                           [0.0101225 0.0101225 … -3.02
                                                           [0.0107273 0.0107273 … -4.52
                                                           [0.0113008 0.0113008 … -6.72
                                                           [0.0113296 0.0113296 … -5.45
```

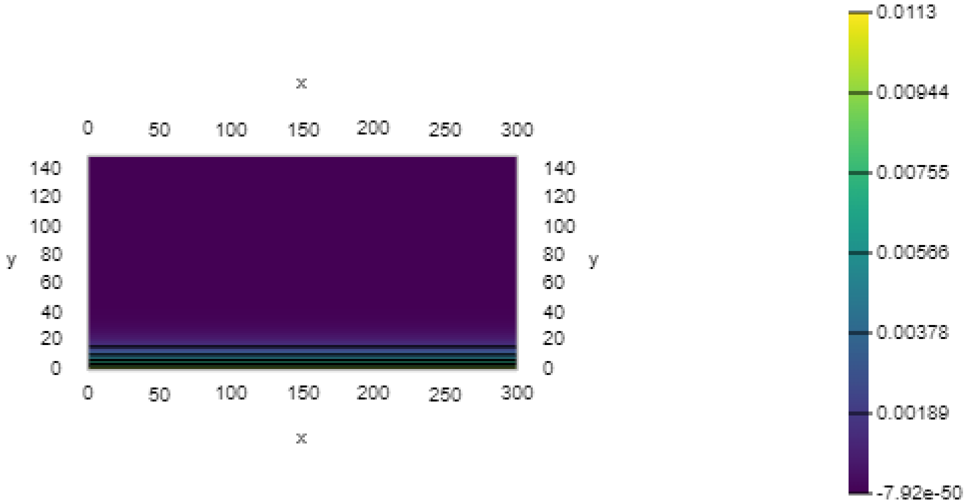
```julia
begin
    tend = 10 #final time
    steadystate = false #solving for steady state or transient depending on boolean
value
    grid,sol,nf = Porous_medium(68,34,10,steadystate,tend)
end
```

t=

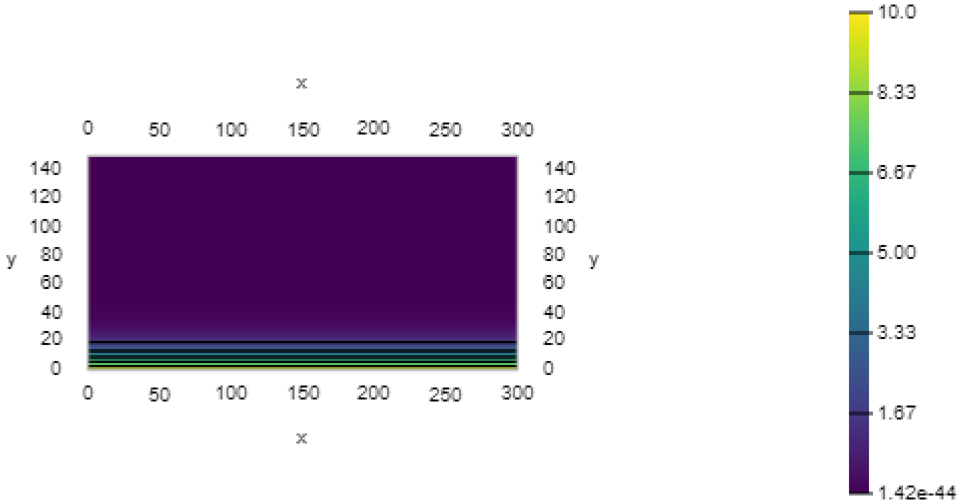
```julia
if steadystate
else
    md"""
    t= $(@bind t_plot Slider(0:tend/10000:tend,default=0))
    """
end
```

```
2×2312 Matrix{Float64}:
  0.0113296    0.0113296    0.0113296    0.0113296   …  -5.45398e-50   -5.43986e-50
 10.0         10.0         10.0         10.0              1.42092e-44    1.42092e-44
```

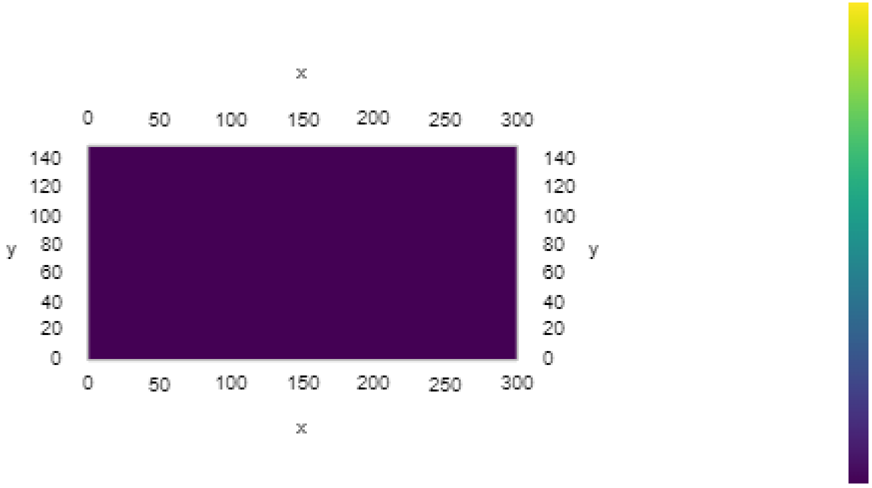
```julia
if steadystate
    tsol = sol
else
    tsol = sol(t_plot);
end
```



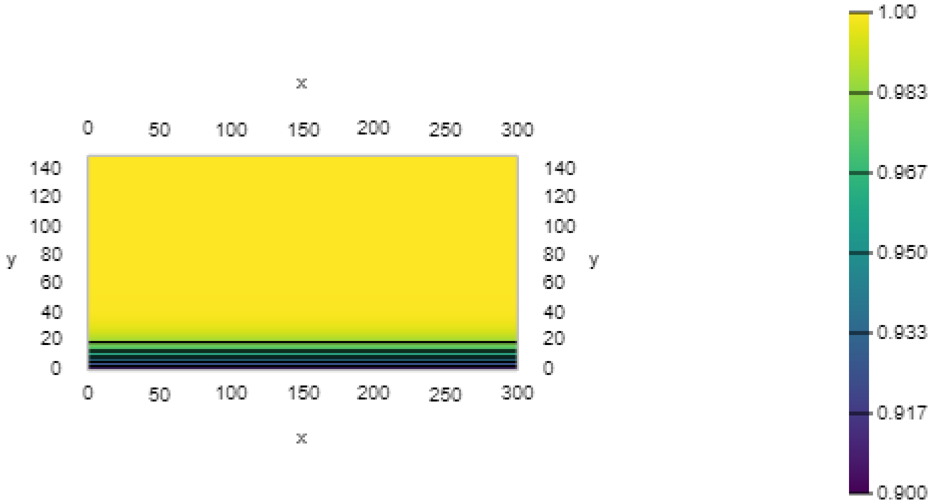

```
begin#plotting pressure
```




```
begin#plotting temperature
```




```
begin#plotting q in x direction
    vis3=GridVisualizer(size=(600,300),xlabel="x",legend=:rt);vis3
    scalarplot!(vis3,grid,-ρ_ref*k*nf[1,1,:],color=:red,label="u_2")
    reveal(vis3)
end
```




```
begin#plotting q in y direction
```