```
using JuMP
using HiGHS

# Model
model = Model(HiGHS.Optimizer)

# Decision variables
@variable(model, x[1:2, 1:3] >= 0)  # Amount of Crude-1 and Crude-2 used in months (in thousands of barrels)
@variable(model, g[1:2, 1:3] >= 0)  # Amount of Gas-1 and Gas-2 produced (in thousands of barrels)
@variable(model, s[1:2, 1:3] >= 0)  # Amount of Gas-1 and Gas-2 stored (in thousands of barrels)

# Parameters
demand = [60 70; 45 80; 90 75]  # Demand for Gas-1 and Gas-2 (in thousands of barrels)
cost_crude = [75000 72000; 83000 75000; 64000 70000]  # Costs for Crude-1 and Crude-2 (in dollars per thousand barrels)
availability = 95  # available crude oil per month (in thousands of barrels)
holding_cost = 3000  # Holding cost per thousand barrels
degradation_rate = 0.02  # Degradation rate??

# Objective min
@objective(model, Min,
    sum(cost_crude[j, i] * x[i, j] for i in 1:2 for j in 1:3) +  # Crude oil costs
    sum(holding_cost * s[i, j] for i in 1:2 for j in 1:3)       # Holding costs
)

# Constraints
# Production capacity constraints
@constraint(model, g[1, :] + g[2, :] .<= 150)  # Total production limit for gasoline

# Demand with degradation for each gasoline type
@constraint(model, s[1, 1] .== g[1, 1] .- demand[1, 1])  # Month 1 for Gas-1
@constraint(model, s[2, 1] .== g[2, 1] .- demand[1, 2])  # Month 1 for Gas-2
@constraint(model, s[1, 2] .== g[1, 2] .+ s[1, 1] .* (1 - degradation_rate) .- demand[2, 1])  # Month 2 for Gas-1
@constraint(model, s[2, 2] .== g[2, 2] .+ s[2, 1] .* (1 - degradation_rate) .- demand[2, 2])  # Month 2 for Gas-2
@constraint(model, s[1, 3] .== g[1, 3] .+ s[1, 2] .* (1 - degradation_rate) .- demand[3, 1])  # Month 3 for Gas-1
@constraint(model, s[2, 3] .== g[2, 3] .+ s[2, 2] .* (1 - degradation_rate) .- demand[3, 2])  # Month 3 for Gas-2
```

```julia
# Crude oil usage constraints
@constraint(model, g[1, :] .<= 12 .* x[1, :] .+ 12 .* x[2, :])  # Gas-1 production limits
@constraint(model, g[2, :] .<= 8 .* x[1, :] .+ 8 .* x[2, :])    # Gas-2 production limits

# Availability constraint
@constraint(model, x.<= availability)  # crude oil availability

# Solve
optimize!(model)

# Results
optimal_value = objective_value(model)
decision_variables = value.(x)
gas_produced = value.(g)
gas_stored = value.(s)

println("Optimal objective value: ", optimal_value)
println("Decision variables (Crude oil usage): ")
println(decision_variables)
println("Gas produced: ")
println(gas_produced)
println("Gas stored: ")
println(gas_stored)
```

```
Running HiGHS 1.7.2 (git hash: 5ce7a2753): Copyright (c) 2024 HiGHS
under MIT licence terms
Coefficient ranges:
  Matrix [1e+00, 1e+01]
  Cost   [3e+03, 8e+04]
  Bound  [0e+00, 0e+00]
  RHS    [4e+01, 2e+02]
Presolving model
13 rows, 16 cols, 36 nonzeros  0s
13 rows, 16 cols, 36 nonzeros  0s
Presolve : Reductions: rows 13(-8); columns 16(-2); elements 36(-10)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     3.3395238579e+00 Pr: 6(420) 0s
         10     2.0259183673e+06 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 10
Objective value     :  2.0259183673e+06
HiGHS run time      :          0.00
Optimal objective value: 2.0259183673469387e6
Decision variables (Crude oil usage):
```

```
[0.0 0.0 9.375; 8.75 10.0 0.0]
Gas produced:
[60.0 60.30612244897959 75.0; 70.0 80.0 75.0]
Gas stored:
[0.0 15.306122448979592 0.0; 0.0 0.0 0.0]
```