In [1]:

```
###To import the necessary libaries

import pandas as pd
import numpy as np
from gurobipy import *
```

In [2]:

```
#########Parameters Set-up############

# Production, budget and others
max_production = 96000
min_production = [0]+[60000]*12
max_production_budget = 673719
fcr = 2.2

# Cost
h = [0.036]*13
var_cost = [0]+ [0.2051444222]*12
f = 13260.66667

# Prices for the different feed
pe_one = [0, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105, 0.105]
pe_two = [0, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115]

# Price for the catfishes
p_one = [0, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]
#Wald (Pessimistic)
#p_two =[0, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]
#Laplace
#p_two = [0, 0.70665, 0.7852, 0.7852, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]
#Max regret
#p_two = [0, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.8380, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]
#Hurwicz
#p_two = [0, 0.70665, 0.70665, 0.8441, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]
#Benefit
#p_two = [0, 0.8456, 0.70665, 0.8456, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]
#Wald (Optimistic)
p_two = [0, 0.70665, 0.70665, 0.8901, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665, 0.70665]

#survival rate
sr = [0.8490,0.8490]
#sr = [0.8490, 0.9052]

t= len(min_production)

print("min_production:", len(min_production))
print("h:", len(h))
print("var_cost:", len(var_cost))
print("pe_one:", len(pe_one))
print("pe_two:", len(pe_two))
print("p_one:", len(p_one))
```

```
print("p_two:", len(p_two))
print("t:", t)
```

```
min_production: 13
h: 13
var_cost: 13
pe_one: 13
pe_two: 13
p_one: 13
p_two: 13
t: 13
```

In [3]:

```
###Without uncertainty to the survival rate for the feed
```

In [4]:

```
#########Model Set-up###############

m = Model("production")

### Decision Variables:
# q = quantity of catfish produced (in pound) in each month t
q_one = m.addVars(t, name = "quantity_produced_feedone")
q_two = m.addVars(t, name = "quantity_produced_feedtwo")
# x = quantity of catfish (in pound) that will be kept as inventory in each month t
x = m.addVars(t, name = "inventory_kept")
# dt = quantity of catfish (in pound) that will be sold to the intermediaries
d = m.addVars(t, name = "quantity_sold")

# e_one = quantity of feed 1 used in each month t (pound/month) (except month 0 for the planning month)
e_one = m.addVars(t, name = "quantity_feed_one")
# e_two = quantity of feed 2 used in each month t (pound/month) (except month 0 for the planning month)
e_two = m.addVars(t, name = "quantity_feed_two")
```

```
Academic license - for non-commercial use only - expires 2021-12-11
Using license file C:\Users\Sophil\gurobi.lic
```

In [5]:

```
#to set the objective function
m.setObjective( ( quicksum(p_one[i]*min_production[i] for i in range(t))
                 + quicksum(np.max((d[i]- min_production[i]),0) * p_two[i] for i in range(t))
                 -12*f
                 - quicksum(var_cost[i]*(q_one[i] + q_two[i]) for i in range(t))
                - quicksum(pe_one[i]*e_one[i] for i in range(t))
                - quicksum(pe_two[i]*e_two[i] for i in range(t)) - quicksum(h[i]*x[i] for i in range(t)) ), GRB.MAXIMIZE)
```

In [6]:

```
#Add the constraints for start, t=0 and t=12
m.addConstr(q_one[0] == 0, "quantity produced at the start, t=0")
m.addConstr(q_two[0] == 0, "quantity produced at the start, t=0")
m.addConstr(x[0] == 60000, "inventory at the start, t=0")
m.addConstr(d[0] == 0 , "quantity sold at the start, t=0")
m.addConstr(e_one[0] == 0, "quantity of feed 1 at the start, t=0")
m.addConstr(e_two[0] == 0, "quantity of feed 2 at the start, t=0")
```

```
m.addConstr(x[12] >= 60000, "inventory at the end, t=12")

###Add the inventory constraint
m.addConstr( ( (x[1] == sr[0]*q_one[1] + sr[1]*q_two[1] + x[0] - d[1]) ) , "inventory1")
m.addConstr( ( (x[2] == sr[0]*q_one[2] + sr[1]*q_two[2] + x[1] - d[2]) ) , "inventory2")
m.addConstr( ( (x[3] == sr[0]*q_one[3] + sr[1]*q_two[3] + x[2] - d[3]) ) , "inventory3")
m.addConstr( ( (x[4] == sr[0]*q_one[4] + sr[1]*q_two[4] + x[3] - d[4]) ) , "inventory4")
m.addConstr( ( (x[5] == sr[0]*q_one[5] + sr[1]*q_two[5] + x[4] - d[5]) ) , "inventory5")
m.addConstr( ( (x[6] == sr[0]*q_one[6] + sr[1]*q_two[6] + x[5] - d[6]) ) , "inventory6")
m.addConstr( ( (x[7] == sr[0]*q_one[7] + sr[1]*q_two[7] + x[6] - d[7]) ) , "inventory7")
m.addConstr( ( (x[8] == sr[0]*q_one[8] + sr[1]*q_two[8] + x[7] - d[8]) ) , "inventory8")
m.addConstr( ( (x[9] == sr[0]*q_one[9] + sr[1]*q_two[9] + x[8] - d[9]) ) , "inventory9")
m.addConstr( ( (x[10] == sr[0]*q_one[10] + sr[1]*q_two[10] + x[9] - d[10]) ) , "inventory10")
m.addConstr( ( (x[11] == sr[0]*q_one[11] + sr[1]*q_two[11] + x[10] - d[11]) ) , "inventory11")
m.addConstr( ( (x[12] == sr[0]*q_one[12] + sr[1]*q_two[12] + x[11] - d[12]) ) , "inventory12")

# Add selling quantity constraint (dt >= mt)
m.addConstrs( ( d[i]  >= min_production[i] for i in range(t) ) ,"Selling_quantity")

#Add production capacity production
m.addConstrs( ( q_one[i] + q_two[i] <= max_production for i in range(t) ) ,"production_capacity")

#Add budget for feed constraint ((∑ pe_onet * e_onet  + ∑ pe_twot * e_twot)
# <= max_production_budget - (12 * ft) - (∑ vt*qt)) - (∑ ht*xt)
m.addConstr( ( quicksum(pe_one[i]*e_one[i] + pe_two[i]*e_two[i] for i in range(t))
              <= max_production_budget - 12*f -quicksum(var_cost[i]*(q_one[i] + q_two[i]) for i in range(t))
                 - quicksum(x[i]*h[i] for i in range(t)) ), "feed_budget")

#Add feed constraint (e_onet  + e_twot >= FCR * qt)
m.addConstrs( ( e_one[i]/ 2.2 >= q_one[i] for i in range(t) ), "feed_constraint1")
m.addConstrs( ( e_two[i]/ 2.2 >= q_two[i] for i in range(t) ), "feed_constraint2")
#m.addConstrs( ( (e_one[i] + e_two[i]) >= 2.2*q[i] for i in range(t) ) ,"feed_constraint")

#Add survival constraint (0.8490 e_onet +  0.97052 e_twot - 0.8(e_onet  + e_twot) >= 0)
m.addConstrs( ( sr[0]*e_one[i] + sr[1]*e_two[i]  >= 0.8*(e_one[i] + e_two[i]) for i in range(t) ), 'Survival')
```

Out[6]:
```
{0: <gurobi.Constr *Awaiting Model Update*>,
 1: <gurobi.Constr *Awaiting Model Update*>,
 2: <gurobi.Constr *Awaiting Model Update*>,
 3: <gurobi.Constr *Awaiting Model Update*>,
 4: <gurobi.Constr *Awaiting Model Update*>,
 5: <gurobi.Constr *Awaiting Model Update*>,
 6: <gurobi.Constr *Awaiting Model Update*>,
 7: <gurobi.Constr *Awaiting Model Update*>,
 8: <gurobi.Constr *Awaiting Model Update*>,
 9: <gurobi.Constr *Awaiting Model Update*>,
 10: <gurobi.Constr *Awaiting Model Update*>,
 11: <gurobi.Constr *Awaiting Model Update*>,
 12: <gurobi.Constr *Awaiting Model Update*>}
```

In [7]:
```
# Solving the model
m.optimize()
```

```python
#  Print optimal solutions and optimal value
print("\n Optimal Solution :\n")
for i, v in enumerate(m.getVars()):
    print(v.VarName, v.x)

print('Obj:', m.objVal)
```

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 85 rows, 78 columns and 245 nonzeros
Model fingerprint: 0x1aa73404
Coefficient statistics:
  Matrix range     [4e-02, 1e+00]
  Objective range  [4e-02, 9e-01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [6e+04, 5e+05]
Presolve removed 60 rows and 31 columns
Presolve time: 0.01s
Presolved: 25 rows, 47 columns, 117 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    6.9306000e+31   1.200000e+31   6.930600e+01      0s
      35    3.9149748e+04   0.000000e+00   0.000000e+00      0s

Solved in 35 iterations and 0.01 seconds
Optimal objective  3.914974804e+04

 Optimal Solution :

quantity_produced_feedone[0] 0.0
quantity_produced_feedone[1] 96000.0
quantity_produced_feedone[2] 96000.0
quantity_produced_feedone[3] 96000.0
quantity_produced_feedone[4] 96000.0
quantity_produced_feedone[5] 96000.0
quantity_produced_feedone[6] 96000.0
quantity_produced_feedone[7] 96000.0
quantity_produced_feedone[8] 96000.0
quantity_produced_feedone[9] 94148.8370060367
quantity_produced_feedone[10] 96000.0
quantity_produced_feedone[11] 96000.0
quantity_produced_feedone[12] 96000.0
quantity_produced_feedtwo[0] 0.0
quantity_produced_feedtwo[1] 0.0
quantity_produced_feedtwo[2] 0.0
quantity_produced_feedtwo[3] 0.0
quantity_produced_feedtwo[4] 0.0
quantity_produced_feedtwo[5] 0.0
quantity_produced_feedtwo[6] 0.0
quantity_produced_feedtwo[7] 0.0
quantity_produced_feedtwo[8] 0.0
quantity_produced_feedtwo[9] 0.0
quantity_produced_feedtwo[10] 0.0
quantity_produced_feedtwo[11] 0.0
quantity_produced_feedtwo[12] 0.0
inventory_kept[0] 60000.0
```

```
inventory_kept[1] 81504.0
inventory_kept[2] 103008.0
inventory_kept[3] 0.0
inventory_kept[4] 0.0
inventory_kept[5] 0.0
inventory_kept[6] 0.0
inventory_kept[7] 0.0
inventory_kept[8] 0.0
inventory_kept[9] 0.0
inventory_kept[10] 16992.0
inventory_kept[11] 38496.0
inventory_kept[12] 60000.0
quantity_sold[0] 0.0
quantity_sold[1] 60000.0
quantity_sold[2] 60000.0
quantity_sold[3] 184512.0
quantity_sold[4] 81504.0
quantity_sold[5] 81504.0
quantity_sold[6] 81504.0
quantity_sold[7] 81504.0
quantity_sold[8] 81504.0
quantity_sold[9] 79932.36261812516
quantity_sold[10] 64512.0
quantity_sold[11] 60000.0
quantity_sold[12] 60000.0
quantity_feed_one[0] 0.0
quantity_feed_one[1] 211200.0
quantity_feed_one[2] 211200.0
quantity_feed_one[3] 211200.0
quantity_feed_one[4] 211200.0
quantity_feed_one[5] 211200.0
quantity_feed_one[6] 211200.0
quantity_feed_one[7] 211200.0
quantity_feed_one[8] 211200.0
quantity_feed_one[9] 207127.44141328076
quantity_feed_one[10] 211200.0
quantity_feed_one[11] 211200.0
quantity_feed_one[12] 211200.0
quantity_feed_two[0] 0.0
quantity_feed_two[1] 0.0
quantity_feed_two[2] 0.0
quantity_feed_two[3] 0.0
quantity_feed_two[4] 0.0
quantity_feed_two[5] 0.0
quantity_feed_two[6] 0.0
quantity_feed_two[7] 0.0
quantity_feed_two[8] 0.0
quantity_feed_two[9] 0.0
quantity_feed_two[10] 0.0
quantity_feed_two[11] 0.0
quantity_feed_two[12] 0.0
Obj: 39149.74804409826
```

In [8]:

```
solution = np.array(m.x)
len(solution)
solution = solution.reshape(6,13)
df = pd.DataFrame({'q_one':solution[0],
```

```python
                  'q_two':solution[1],
                  'x':solution[2],
                  'd':solution[3],
                  'e_one':solution[4],
                  'e_two':solution[5]})
df
```

Out[8]:

	q_one	q_two	x	d	e_one	e_two
0	0.000000	0.0	60000.0	0.000000	0.000000	0.0
1	96000.000000	0.0	81504.0	60000.000000	211200.000000	0.0
2	96000.000000	0.0	103008.0	60000.000000	211200.000000	0.0
3	96000.000000	0.0	0.0	184512.000000	211200.000000	0.0
4	96000.000000	0.0	0.0	81504.000000	211200.000000	0.0
5	96000.000000	0.0	0.0	81504.000000	211200.000000	0.0
6	96000.000000	0.0	0.0	81504.000000	211200.000000	0.0
7	96000.000000	0.0	0.0	81504.000000	211200.000000	0.0
8	96000.000000	0.0	0.0	81504.000000	211200.000000	0.0
9	94148.837006	0.0	0.0	79932.362618	207127.441413	0.0
10	96000.000000	0.0	16992.0	64512.000000	211200.000000	0.0
11	96000.000000	0.0	38496.0	60000.000000	211200.000000	0.0
12	96000.000000	0.0	60000.0	60000.000000	211200.000000	0.0

In [ ]: