In [10]:
```
from gamspy import (
    Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
    Domain,Number,Sum,Product,Smax,Smin,Ord,Card,ModelStatus,SpecialValues
)

options = Options(variable_listing_limit=0, equation_listing_limit=100)
m = Container(load_from='goblet.gdx',options=options)
d, data, G, item, T = m.getSymbols(["d","data","G","item","T"])
display(d.pivot(), data.pivot())
```

	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12
g1	20.0	22.0	18.0	35.0	17.0	19.0	23.0	20.0	29.0	30.0	28.0	32.0
g2	17.0	19.0	23.0	20.0	11.0	10.0	12.0	34.0	21.0	23.0	30.0	12.0
g3	18.0	35.0	17.0	10.0	9.0	21.0	23.0	15.0	10.0	0.0	13.0	17.0
g4	31.0	45.0	24.0	38.0	41.0	20.0	19.0	37.0	28.0	12.0	30.0	37.0
g5	23.0	20.0	23.0	15.0	10.0	22.0	18.0	30.0	28.0	7.0	15.0	10.0
g6	22.0	18.0	20.0	19.0	18.0	35.0	0.0	28.0	12.0	30.0	21.0	23.0

	g1	g2	g3	g4	g5	g6
Production_Cost	100.0	80.0	110.0	90.0	200.0	140.0
Holding_Cost	25.0	28.0	25.0	27.0	10.0	20.0
Initial_Stock	50.0	20.0	0.0	15.0	0.0	10.0
Final_Stock	10.0	10.0	10.0	10.0	10.0	10.0
Elf_Time	3.0	3.0	3.0	2.0	4.0	4.0
Machine_Time	2.0	1.0	4.0	8.0	11.0	9.0
Storage_Space	4.0	5.0	5.0	6.0	4.0	9.0

In [68]:
```
I = Variable(m, 'I', 'positive', domain=[G, T], description='Leftover invent
P = Variable(m, 'P', 'positive', domain=[G, T], description='Each Type of Go
S = Variable(m,'S','positive',[G, T],description='Shortage inventory in scer
# L = Variable(m,'L','positive',[G, T],description='Leftover inventory in sc


# theta = Parameter(m,'theta',description='Backlog cost',records=0.8)

BalShoe_eq = Equation(m,'BalShoe_eq',domain=[G, T])
BalShoe_eq[G, T]= I[G, T]== data["Initial_Stock", G].where[T.first] + I[G, T

elfTime = Equation(m, 'elfTime', domain=[G, T])
elfTime[G, T] = data['Elf_Time', G] <= 420

machineTime = Equation(m, 'machineTime', domain=[G, T])
machineTime[G, T] = data['Machine_Time', G] <= 800
```

```
storageSpace = Equation(m, 'Storage_Space', domain=[G])
storageSpace[G] = data['Storage_Space', G] <= 1000

demand = Equation(m, 'demand', domain=[G, T])
demand[G, T] = P[G, T] + I[G, T] >= d[G, T]

backlog = Equation(m, 'backlog', domain=[G, T])
backlog[G, T] = S[G, T] >= 0

BacklogDef_eq = Equation(m,'BacklogDef_eq', domain=[G, T])
BacklogDef_eq[G, T]=  data['Final_Stock', G] == L[G, T] - S[G, T]

backlog = Model(m,"goblet",
    equations=m.getEquations(),
    problem=Problem.LP,
    sense=Sense.MIN,
    objective=Sum([G, T], I[G, T] * data["Holding_Cost", G] + P[G, T] * data
)

I.lo[G, T] = SpecialValues.NEGINF
I.lo[G, T].where[T.last] = 0

backlog.solve(options=Options(relative_optimality_gap=1e-6))
```

Out[68]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solve
0	Normal	OptimalGlobal	189280.294189453	289	289	LP	CPLE

In [69]:
```
display(backlog.objective_value)
display(I.pivot())
```

189280.29418945312

	w1	w2	w3	w4	w5	w6	w7	w8	w9	
g1	30.0	15.00	7.500	3.7500	1.87500	0.937500	0.468750	0.234375	0.117188	0
g2	10.0	5.00	2.500	1.2500	0.62500	0.312500	0.156250	0.078125	0.039062	0
g3	0.0	0.00	0.000	0.0000	0.00000	0.000000	0.000000	0.000000	0.000000	0
g4	7.5	3.75	1.875	0.9375	0.46875	0.234375	0.117188	0.058594	0.029297	0
g5	0.0	0.00	0.000	0.0000	0.00000	0.000000	0.000000	0.000000	0.000000	0
g6	5.0	2.50	1.250	0.6250	0.31250	0.156250	0.156250	0.078125	0.039062	0

In [ ]: