### DATA

1. *2025-07-14\_MDRH\_Inventory\_Storage\_Burn\_Rates\_V3.xlsx*   
   we care about two tabs

**01. Data (Department Rollup).** (Provides lead times, and the map of which SKUs are stored in which inventories)

**02. Full Data** (Provides demand data)

1. *2025-08-04\_MDRH\_Inventory\_Safety\_Stock\_Sample\_Items.xlsx*

Provides client’s already calculated target inventories. (AKA analytical solution)

### Specifications

1. One safety stock (AKA Perpetual Inventory)
2. 17 cycle stocks (AKA PARs)
3. Each inventory replenishes independently, but if there is a stockout at PAR level, we will grab material from the safety stock.

A diagram of a diagram

AI-generated content may be incorrect.

## Steps:

1. using historical demand data to calculate stockouts (just one year)

A graph of a period of time

AI-generated content may be incorrect.

1. implementing client’s target inventories to compute their stockouts.

The PAR inventories should face the demand for that day, if they have enough, they would provide it. This will initiate an order to be delivered after the lead time for that SKU.

If PARs don’t have enough SKU to supply the demand, PAR stockout happens. PAR stockouts will be covered by the Perpetual as much as it can. The remaining unsupplied demand would become hospital level stockouts.

A diagram of a company's supply chain

AI-generated content may be incorrect.

A diagram of a process

AI-generated content may be incorrect.

Main Equations:

* *Inventory Gap= MAX(0, ((depleting\*DT+target\_inventory)-(SKUs\_in\_Shipment+PAR)))*

The first part is what we want, the second part is what we already own. The difference shows the gap we need to order.

* *PAR Stockout = (demand\_projection-depleting\*DT/day)*

PARs cannot go negative, therefore *depleting* is automatically bound to whatever is in the PAR. If it is less than the demand projection, PAR stockouts gets value.

* *Transit time for SKU in Shipment = Lead Time*
* *Perpetual.supplying\_PAR = SUM(PAR.PAR\_stockouts[itemType,\*])\*day/DT*

At each time step, Perpetual supplies the combined PAR level stockouts for each SKU

* *Supplying from Perpetual = ALLOCATE(Perpetual.supplying\_PAR[itemType]\*DT/day, PARInventory, PAR\_stockouts[itemType,\*], PAR\_priority[itemType,\*], 0)*

In ALLOCATE function:

* first argument is what to be allocated (SKUs coming from perpetual)
* second argument is the category for allocating (the PAR inventories)
* third argument is the amount each PAR needs. (PAR Stockouts)
* fourth argument is the priority given to each PAR (it is 1 for all PARs in this case)
* fifth argument is the spread of distribution (If **priority\_spread** is 0 then higher priority indices are supplied first. When it is positive, higher priority indices will get a larger share or their target, but lower priority indices may also receive a portion of their target)

1. Using King’s method to recalculate the target inventories and comparing them with the client’s results

A diagram of a company

AI-generated content may be incorrect.

The equation they are supposedly using  
A math formula with black text

AI-generated content may be incorrect.  
  
Desired Service level is 98% meaning z-core 2.05

Historical Standard Deviation should be calculated based on demand data.   
  
For Cycle Stocks, they have used the same equation above.   
I have used the Demand MEAN per occurrence.

The average is suggested by King’s cycle stock equations.

Daily average would result in very low stocks because the data is not following normal distribution. So, I changed it to MEAN per occurrence.

### Cautions

* The given data does not show variability in lead time. But it seems that they have data for that and use that for their calculation. This is mentioned in the last meeting and the last meeting only. The data for variability was never delivered.
* It seems that they have a policy to store SKUs for at least 2 days. Again, this was briefly mentioned only in the last meeting. I am not sure what their real policy is.
* They care about frequency of stockouts in one year, rather than the accumulative volume of stockouts.

### Equations

|  |  |  |  |
| --- | --- | --- | --- |
|  | Equation | Properties | Units |
| Perpetual: | | | |
| Perpetual[itemType](t) | Perpetual[itemType](t - dt) + (replenishing[itemType] - supplying\_PAR[itemType]) \* dt | INIT Perpetual[itemType] = 0 | Each |
| SKUs\_in\_Shipment[itemType](t) | SKUs\_in\_Shipment[itemType](t - dt) + (ordering[itemType] - replenishing[itemType]) \* dt | INIT SKUs\_in\_Shipment[itemType] = 0 TRANSIT TIME = lead\_time CONTINUOUS ACCEPT MULTIPLE BATCHES | Each |
| ordering[itemType] | (inventory\_gap+STEP(safety\_stock\*day, ordering\_for\_the\_new\_building\_time, DT))/DT |  | Each/Days |
| replenishing[itemType] | CONVEYOR OUTFLOW |  | Each/Days |
| supplying\_PAR[itemType] | SUM(PAR.PAR\_stockouts[itemType,\*])\*day/DT |  | Each/Days |
| burn\_rate[itemType] | IF TIME < current\_time THEN RUNMEAN(SUM(PAR.historical\_demand[itemType,\*])) ELSE PREVIOUS(SELF, RUNMEAN(SUM(PAR.historical\_demand[itemType,\*])) ) |  | Each/days |
| current\_time | 518 |  | Days |
| day | 1 |  | Days |
| desired\_cycle\_service\_level[itemType] | GRAPH(strategic\_importance) Points: (1.000, 0.840), (2.000, 0.850), (3.000, 0.900), (4.000, 0.950), (5.000, 0.970), (6.000, 0.980), (7.000, 0.990), (8.000, 0.999) |  | dmnl |
| historical\_demand\_standard\_deviation[itemType] | IF TIME < current\_time THEN (sum\_squared\_deviations//(TIME-1))^0.5 ELSE PREVIOUS(SELF, (sum\_squared\_deviations//(TIME-1))^0.5) |  | Each/Days |
| inventory\_gap[itemType] | MAX(0, ((supplying\_PAR\*DT+target\_inventory)-(SKUs\_in\_Shipment+Perpetual))) |  | Each |
| lead\_time["00136"] | 0.5 |  | Days |
| lead\_time["00235"] | 0.34 |  |  |
| lead\_time["00262"] | 7.5 |  |  |
| lead\_time["00386"] | 0.5 |  |  |
| lead\_time["00508"] | 0.66 |  |  |
| lead\_time["00523"] | 0.5 |  |  |
| lead\_time["00529"] | 0.5 |  |  |
| lead\_time["00535"] | 0.34 |  |  |
| lead\_time["00536"] | 0.66 |  |  |
| lead\_time["00549"] | 0.5 |  |  |
| lead\_time["00552"] | 1 |  |  |
| lead\_time["00555"] | 0.5 |  |  |
| lead\_time["00565"] | 0.66 |  |  |
| lead\_time["00838"] | 0.5 |  |  |
| lead\_time["01077"] | 3 |  |  |
| lead\_time["30847"] | 0.34 |  |  |
| lead\_time["698758"] | 1 |  |  |
| lead\_time["698770"] | 1 |  |  |
| lead\_time["803949"] | 1.5 |  |  |
| lead\_time["803975"] | 1 |  |  |
| lead\_time["803992"] | 1 |  |  |
| lead\_time["804039"] | 1 |  |  |
| lead\_time["804046"] | 1.5 |  |  |
| lead\_time["804052"] | 1 |  |  |
| ordering\_for\_the\_new\_building\_time | 620 |  | Day |
| safety\_stock[itemType] | "z-score"\* historical\_demand\_standard\_deviation\* ((lead\_time)/day)^0.5 |  | Each/Days |
| strategic\_importance[itemType] | 6 |  | dmnl |
| sum\_squared\_deviations[itemType] | IF TIME < current\_time THEN RUNCUM((SUM(PAR.historical\_demand[itemType,\*])-burn\_rate)^2) ELSE PREVIOUS(SELF, RUNCUM((SUM(PAR.historical\_demand[itemType,\*])-burn\_rate)^2)) |  | Each^2/Days |
| target\_inventory[itemType] | IF TIME < PAR.new\_building\_start\_day THEN 0 ELSE safety\_stock\*day |  | Each |
| "z-score"[itemType] | GRAPH(desired\_cycle\_service\_level\*100) Points: (84.00, 1.000), (85.00, 1.040), (90.00, 1.280), (95.00, 1.650), (97.00, 1.880), (98.00, 2.050), (99.00, 2.330), (99.90, 3.090) |  | dmnl |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Equation | Properties | Units |
| PAR: | | | |
| Items\_in\_Shipment[itemType, PARInventory](t) | Items\_in\_Shipment[itemType, PARInventory](t - dt) + (ordering[itemType, PARInventory] - replenishing[itemType, PARInventory]) \* dt | INIT Items\_in\_Shipment[itemType, PARInventory] = 0 TRANSIT TIME = Perpetual.lead\_time[itemType] CONTINUOUS ACCEPT MULTIPLE BATCHES | Each |
| PAR[itemType, PARInventory](t) | PAR[itemType, PARInventory](t - dt) + (replenishing[itemType, PARInventory] - depleting[itemType, PARInventory]) \* dt | INIT PAR[itemType, PARInventory] = 0 | Each |
| depleting[itemType, PARInventory] | demand\_projection\*day/DT |  | Each/Days |
| ordering[itemType, PARInventory] | (inventory\_gap+STEP(initial\_PAR, ordering\_for\_the\_new\_building\_time, DT))/DT |  | Each/Days |
| replenishing[itemType, PARInventory] | CONVEYOR OUTFLOW |  | Each/Days |
| burn\_rate[itemType, PARInventory] | RUNCUM(historical\_demand)//RUNCUM(IF historical\_demand > 0 THEN 1 ELSE 0)/day |  | Each/Days |
| day | 1 |  | Days |
| demand\_projection[itemType, PARInventory] | DELAY(historical\_demand, new\_building\_start\_day, 0) |  | Each/Day |
| fill\_rate[itemType, PARInventory] | (1-RUNCUM(PAR\_stockouts)//RUNCUM(demand\_projection))\*100 |  | Percentage |
| frequency\_of\_PAR\_stockouts[itemType, PARInventory] | RUNCUM(IF PAR\_stockouts > 0 THEN 1 ELSE 0) |  | Stockouts |
| frequency\_of\_perpetual\_stockouts[itemType, PARInventory] | RUNCUM(IF PAR\_and\_Perpetual\_stockouts > 0 THEN stockout\_frequency\_unit/DT ELSE 0) |  | Stockout Incident |
| historical\_demand[itemType, PARInventory] | IF ISNAN(total\_quantity\_data) THEN 0 ELSE total\_quantity\_data |  | Each/Day |
| initial\_PAR[itemType, PARInventory] | STEP(burn\_rate\*day, ordering\_for\_the\_new\_building\_time, day) |  | Each |
| inventory\_gap[itemType, PARInventory] | MAX(0, ((depleting\*DT+target\_inventory)-(Items\_in\_Shipment+PAR))) |  | Each |
| new\_building\_start\_day | 650 |  | Day |
| ordering\_for\_the\_new\_building\_time | 620 |  | Day |
| PAR\_and\_Perpetual\_stockouts[itemType, PARInventory] | PAR\_stockouts-supplying\_from\_perpetual |  | Each/Days |
| PAR\_priority[itemType, PARInventory] | 1 |  | dmnl |
| PAR\_stockouts[itemType, PARInventory] | (demand\_projection-depleting\*DT/day) |  | Each/Days |
| stockout\_frequency\_unit | 1 |  | Stockout Incident |
| supplying\_from\_perpetual[itemType, PARInventory] | ALLOCATE(Perpetual.supplying\_PAR[itemType]\*DT/day, PARInventory, PAR\_stockouts[itemType,\*], PAR\_priority[itemType,\*], 0) |  | Each/Days |
| target\_inventory[itemType, PARInventory] | IF TIME < new\_building\_start\_day THEN 0 ELSE burn\_rate\*day |  | Each |
| total\_quantity\_data[itemType, PARInventory] | NAN |  | Each/Day |