March 18, 2026

Layered Logistics Model (LLM)

The right starting point is not a single "food model," but a **layered logistics model** with four linked parts:

1. **Demand / consumption model**
2. **Supply / reserve model**
3. **Transportation network model**
4. **Financial / budget model**

A very good architecture for your case is a **dynamic multi-city, multi-commodity, multi-period network optimization model**. In practice, that usually combines:

- **food balance / demand forecasting**
- **minimum-cost flow or mixed-integer network optimization**
- **inventory / safety stock modeling**
- **vehicle routing or mode-selection optimization**
- **scenario simulation for disruptions and shortages**

Google OR-Tools supports minimum-cost flow and vehicle routing, including capacity and time-window constraints, and Pyomo is a widely used Python optimization framework for building custom linear, mixed-integer, and stochastic models. anyLogistix is a commercial platform that combines network design, optimization, simulation, and digital-twin-style supply chain modeling.

Best model structure

For your problem, I would define the model as:

Sets

- **Cities**: i, j
- **Food products**: k
- **Time periods**: t
- **Transport modes**: m
  truck, rail, ship, air
- **Supply nodes / reserve depots / ports**: s

Core variables

- $D_{i,k,t}$: forecast demand in city $i$for food $k$at time t
- $C_{i,k,t}$: actual consumption
- $P_{s,k,t}$: available supply / production at source s
- $X_{s,i,k,m,t}$: quantity shipped from source $s$to city i

- $Y_{i,j,k,m,t}$: intercity transfer quantity
- $I_{i,k,t}$: ending inventory in city i
- $R_{i,k,t}$: strategic reserve in city i
- $S_{i,k,t}$: shortage / unmet demand
- $Cap_{i,j,m,t}$: transport capacity on link $i \rightarrow j$by mode m
- $Cost_{i,j,m,t}^{trans}$: transport cost per unit
- $Cost_{i,k,t}^{hold}$: inventory holding cost
- $Cost_{i,k,t}^{short}$: shortage penalty
- B_t: budget available
- F_t: funding required

Demand drivers

To make it realistic, demand should depend on:

- population
- demographic mix
- per-capita calorie or basket requirement
- product substitution

- seasonality
- income / price effects
- emergency buffer requirement
- losses / spoilage
- policy rationing rules

Transportation drivers

- road / rail / sea / air availability
- distance and time
- fleet size
- truck / wagon / vessel capacity
- fuel cost
- driver / labor availability
- border / checkpoint delay
- storage / cold-chain compatibility
- route reliability / disruption risk

Financial drivers

- commodity purchase cost
- transport cost
- warehousing cost
- reserve carrying cost
- capital tied in stock
- emergency procurement cost
- shortage social-cost penalty
- subsidy or ration support
- infrastructure bottlenecks

Objective function

A practical objective is:

Minimize total system cost + shortage risk + instability

That means minimizing:

- transport cost
- procurement cost
- storage cost

- reserve cost
- mode-switch cost
- unmet demand penalty
- volatility penalty for low food availability

In plain English:

deliver enough food to each city, keep adequate reserves, stay within transport and budget limits, and minimize both cost and shortage risk.

Dynamic flow logic

A typical inventory balance is:

$$I_{i,k,t} = I_{i,k,t-1} + \text{inflows} - \text{consumption} - \text{spoilage}$$

Shortage is:

$$S_{i,k,t} = \max\left(0, D_{i,k,t} - \text{available food}_{i,k,t}\right)$$

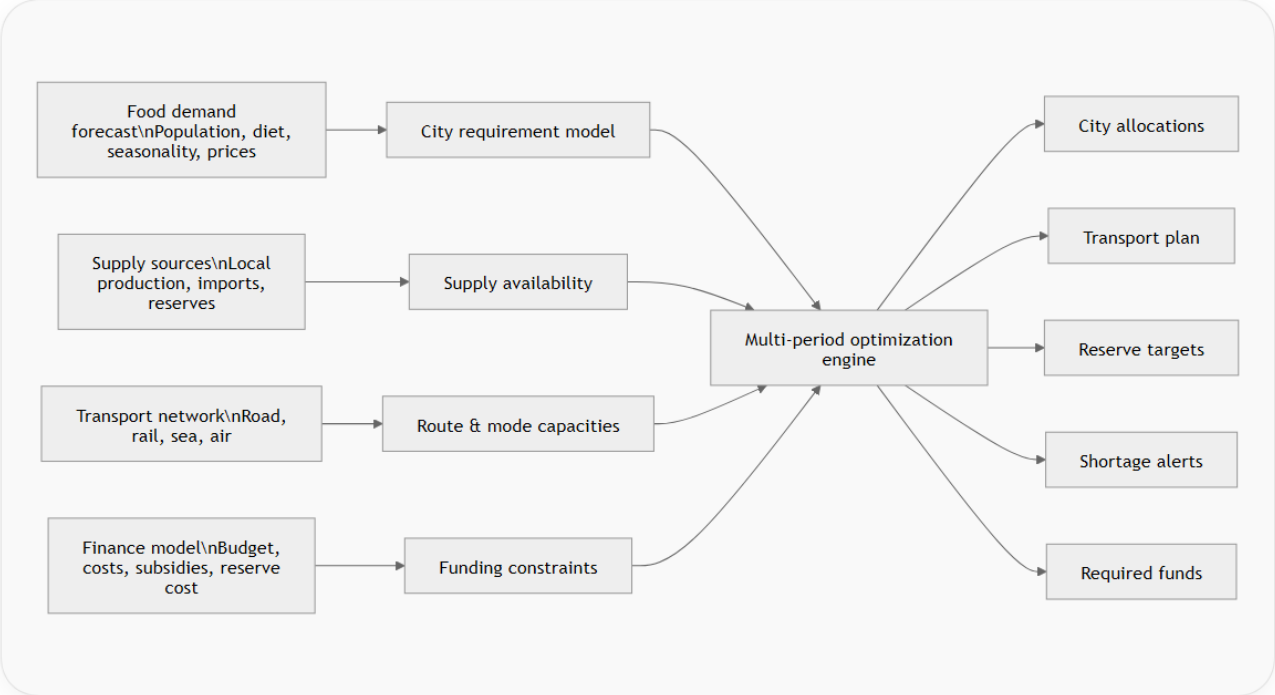
Reserve rule:

$$R_{i,k,t} \geq \alpha_{i,k} \times \text{days of cover target}$$

where αreflects policy such as 7, 14, or 30 days of minimum food cover.

Graphic concept

This is the model shape I recommend:



Best modeling approach by level

Level 1 — fast prototype

Use:

- Excel or Python
- one food basket
- fixed monthly demand
- one transport mode

- one reserve policy

This is good for proving logic.

Level 2 — serious planning model

Use:

- Python + Pyomo or OR-Tools
- multiple cities
- multiple food groups
- monthly periods
- inventory and reserve constraints
- transport capacity and costs
- budget constraints

This is probably the sweet spot for your project. Pyomo is well suited for custom optimization formulations, and OR-Tools provides specialized solvers for network flow and routing.

Level 3 — digital twin / scenario lab

Use:

- anyLogistix or a custom Python stack

- network design + simulation
- disruption scenarios
- facility and depot decisions
- safety stock estimation
- service-level analysis

anyLogistix specifically markets network optimization, inventory planning, simulation, and digital-twin workflows for supply chains.

Recommended tools

Open-source stack

Best for building with me

- **Pyomo** for optimization model structure
- **OR-Tools** for min-cost flow, routing, fleet assignment, and path decisions
- **openrouteservice / OpenStreetMap** for road network, distances, time matrices, and route calculations; openrouteservice provides matrix and optimization services, and OSM data supports routable road networks with tagged map features.

Commercial platform

Best if you want a visual environment quickly

- **anyLogistix** for network design, optimization, simulation, safety stock, and digital twin work.

Good data sources

Food requirement / supply

- **FAOSTAT** gives food and agriculture statistics for more than 245 countries and territories, including food balance and production data.

Commodity / food price inputs

- **World Bank commodity price data (Pink Sheet)** provides recent commodity price series and monthly updates.

Transportation availability / freight structure

- For U.S.-based work, the **BTS Freight Analysis Framework** provides freight flows by origin-destination region, commodity type, and transportation mode, including tonnage, value, and ton-miles.

Minimal data tables you need

To build version 1, I would create these tables:

1. Cities

- city_id
- city_name

- population
- income class
- storage capacity
- reserve target days
- priority score

2. Products

- product_id
- product_name
- calories per unit
- weight per unit
- shelf life
- storage type
- minimum basket share

3. Demand

- city_id
- product_id

- period
- baseline demand
- seasonal factor
- emergency factor
- final demand forecast

4. Sources

- source_id
- source_type
- product_id
- available quantity
- procurement cost
- lead time

5. Network links

- from_node
- to_node
- mode

- distance
- travel time
- capacity
- risk score
- cost per ton-km

6. Fleet / mode availability

- mode
- number available
- payload
- operating cost
- max range
- disruption factor

7. Finance

- period
- budget ceiling
- transport budget

- procurement budget
- reserve budget
- emergency fund

8. Policy / service targets

- minimum days of cover
- max shortage allowed
- priority cities
- protected food classes

The optimization model I recommend for you

If I were designing this with you, I would build it in this order:

Phase A — deterministic planning model

- monthly demand forecast
- fixed network capacities
- min-cost allocation
- reserve constraint
- shortage penalty

Phase B — robust model

- uncertain demand
- uncertain transport availability
- disrupted links
- price shocks
- reserve sufficiency test

Phase C — operational routing

- assign trucks or modes
- daily or weekly routes
- dynamic rerouting when bottlenecks occur

Best starting formula for city requirement

A simple food requirement estimate per city:

$$Req_{i,k,t} = Population_{i,t} \times PerCapitaNeed_k \times SeasonFactor_{i,k,t} \times StressFactor_{i,t} \times LossFactor_{i,k,t}$$

Then reserve requirement:

$$ReserveReq_{i,k,t} = Req_{i,k,t} \times \frac{TargetDays_{i,k}}{PeriodDays_t}$$

Then total need to cover:

$$TotalNeed_{i,k,t} = Req_{i,k,t} + ReserveReq_{i,k,t} - OpeningInventory_{i,k,t}$$

My practical recommendation

For your purposes, the best path is:

- **Python + Pyomo + OR-Tools**
- **FAOSTAT + your own city/population/consumption data**
- **OpenStreetMap / openrouteservice for network distances**
- optional **World Bank price data**
- optional **BTS FAF** if the geography is the United States

That stack is flexible, transparent, and good enough to evolve into a real decision-support model. The commercial shortcut is anyLogistix if you want a faster visual environment.

The next useful step is for me to turn this into a **concrete model blueprint** with:

- input tables
- equations

- dashboard layout
- and a first prototype structure in Excel or Python.

Absolutely. Here is a **concrete model blueprint** you can use as the foundation for a real planning and optimization system.

Model title

Dynamic Multi-City Food Requirement, Distribution, Reserve, and Transportation Optimization Model

This model is designed to answer five central questions at once:

1. How much food each city needs over time
2. How much reserve each city should hold
3. How food should move through the network
4. What transportation assets and funding are required
5. How to minimize shortages, instability, and total cost

---

1. Purpose of the model

The model should support a network of cities where food availability depends on:

- population and consumption needs
- available production, imports, and reserves
- storage capacity
- transportation modes and route capacities
- financial constraints
- disruption, delays, and uncertainty

The model is not just a transport model. It is a **food-security logistics model** with dynamic inventory and budget logic.

---

2. Planning horizon and time step

For the first working version, I recommend:

- **Time unit:** monthly
- **Planning horizon:** 12 months
- **Update cycle:** monthly or weekly
- **Geographic scale:** cities as demand nodes, warehouses/ports/farms as supply nodes

Later, if needed, the same structure can be refined to weekly or daily.

---

3. Model architecture

The blueprint has six linked modules:

Module A — Demand and requirement forecast

Calculates city-level food requirement by product and time period.

Module B — Supply availability

Tracks food available from local production, imports, national reserves, and external suppliers.

Module C — Inventory and reserve logic

Tracks opening stock, inflow, use, spoilage, safety stock, and emergency reserve targets.

Module D — Transportation network

Determines how food flows through the network by mode, route, and capacity.

Module E — Financial and budget model

Tracks procurement, transportation, inventory, reserve, and shortage costs.

Module F — Optimization engine

Chooses shipments, inventory levels, reserves, and route allocations that minimize total cost and shortage risk.

---

4. Core entities

The model should be built around five entity types:

Cities

Demand locations where people consume food.

Supply nodes

Farms, import terminals, ports, processing centers, national warehouses, or donor entry points.

Intermediate nodes

Regional hubs, rail terminals, ports, inland depots.

Products

Food classes such as wheat flour, rice, cooking oil, sugar, legumes, milk powder, and other essential basket items.

Transport modes

Truck, rail, sea, air, river, or mixed mode.

---

5. Minimum data structure

Below is the exact table structure I recommend for version 1.

Table 1 — Cities

Field	Description
city_id	Unique city code
city_name	City name
region	Region or province
population	Current population
pop_growth_rate	Growth assumption
priority_level	Criticality score
storage_capacity_tons	Maximum food storage capacity
cold_storage_capacity_tons	For perishables
reserve_target_days	Desired days of food cover
min_service_level	Minimum percentage of demand to meet

Field	Description
disruption_risk_score	Risk level for the city

Table 2 — Products

Field	Description
product_id	Unique product code
product_name	Food item
unit	Ton, kg, liter, carton
calories_per_unit	For nutritional equivalence
shelf_life_days	Expected shelf life
storage_type	Dry, cold, frozen
priority_class	Staple, essential, secondary
substitution_group	Optional substitution family
loss_rate	Normal spoilage/loss rate

Table 3 — Demand Drivers

Field	Description
city_id	Demand city
product_id	Food item
period	Month or week
per_capita_consumption	Consumption norm
seasonality_factor	Seasonal adjustment
stress_factor	Emergency uplift
rationing_factor	Policy-driven reduction
final_demand	Calculated demand

Table 4 — Supply Sources

Field	Description
source_id	Supply node code
source_name	Supplier / depot / port
source_type	Farm, port, warehouse, import

Field	Description
product_id	Food item
period	Time period
available_quantity	Available quantity
procurement_cost_per_unit	Purchase cost
loading_capacity	Max dispatch quantity
lead_time_days	Normal lead time
reliability_score	Supply confidence

Table 5 — Opening Inventory

Field	Description
node_id	City or depot
product_id	Food item
period	Time period
opening_inventory	Starting stock

Field	Description
reserve_inventory	Stock reserved
usable_inventory	Available for normal distribution

Table 6 — Network Links

Field	Description
from_node	Origin
to_node	Destination
mode	Truck / rail / sea / air
distance_km	Distance
travel_time_days	Lead time
capacity_per_period	Max movement
cost_per_unit	Cost by unit
cost_per_ton_km	Variable cost
fixed_route_cost	Optional

Field	Description
risk_score	Disruption or attack risk
status	Open, limited, closed

Table 7 — Fleet / Mode Availability

Field	Description
mode	Mode name
period	Time period
number_available	Vehicles / assets available
average_payload	Capacity per asset
utilization_limit	Maximum use rate
operating_cost_per_trip	Trip cost
fuel_cost_factor	Fuel sensitivity
maintenance_factor	Availability degradation

Table 8 — Budget and Finance

Field	Description
period	Month / week
procurement_budget	Food purchase funds
transportation_budget	Movement funds
storage_budget	Warehousing funds
reserve_budget	Strategic reserve funds
emergency_budget	Contingency funds
total_budget	Total allowed spending
funding_gap_allowed	Optional tolerance

Table 9 — Policy Parameters

Field	Description
product_id	Food item
minimum_days_cover	Minimum reserve target
shortage_penalty	Penalty cost

Field	Description
holding_cost	Inventory carrying cost
reserve_holding_cost	Strategic reserve carrying cost
disposal_cost	Expiry/disposal penalty
substitution_allowed	Yes / no
max_shortage_pct	Allowed shortage ceiling

Table 10 — Scenario Inputs

Field	Description
scenario_id	Scenario name
period	Time period
disrupted_link	Route impacted
capacity_reduction_pct	Loss of capacity
demand_shock_pct	Demand increase
supply_shock_pct	Supply decrease

Field	Description
price_shock_pct	Procurement increase
fuel_shock_pct	Cost increase

---

6. Decision variables

These are the variables the optimization engine will solve for.

Shipment variables

- (X_{s,i,k,m,t}): quantity shipped from supply node (s) to city (i) of product (k) by mode (m) in period (t)
- (Y_{i,j,k,m,t}): quantity transferred from city or hub (i) to city (j)

Inventory variables

- (I_{i,k,t}): ending inventory at city (i)
- (W_{s,k,t}): ending inventory at supply node (s)

Reserve variables

- (R_{i,k,t}): reserve stock held in city (i)

Shortage variables

- $S_{i,k,t}$: unmet demand in city (i)

Transport use variables

- $U_{i,j,m,t}$: whether a route or mode is used
- $V_{m,t}$: total number of vehicles/assets used by mode

Financial variables

- F_t: total funding required in period (t)
- G_t: funding gap in period (t), if allowed

---

7. Core equations

7.1 Demand equation

For each city, product, and period:

$$
D_{i,k,t} = Pop_{i,t} \times PC_{i,k,t} \times Season_{i,k,t} \times Stress_{i,t} \times Ration_{i,k,t}
$$

Where:

- $Pop_{i,t}$ = population
- $PC_{i,k,t}$ = per-capita need

- (Season_{i,k,t}) = seasonal factor
- (Stress_{i,t}) = shock or emergency multiplier
- (Ration_{i,k,t}) = rationing adjustment

7.2 Inventory balance equation

For each city:

[
I_{i,k,t} = I_{i,k,t-1} + \sum_{s,m} X_{s,i,k,m,t} + \sum_{j,m} Y_{j,i,k,m,t} - D_{i,k,t}^{served} - Loss_{i,k,t}
]

with

[
D_{i,k,t}^{served} = D_{i,k,t} - S_{i,k,t}
]

and

[
Loss_{i,k,t} = \lambda_{i,k} \times \text{average inventory/use}
]

7.3 Reserve requirement equation

$$
R_{i,k,t} \ge \frac{ReserveDays_{i,k}}{DaysInPeriod_t} \times D_{i,k,t}
$$

This enforces minimum strategic stock.

7.4 Usable inventory constraint

$$
I_{i,k,t} \ge R_{i,k,t}
$$

Meaning ending inventory cannot fall below reserve target unless shortage or exception rules allow it.

7.5 Supply availability constraint

$$
\sum_{i,m} X_{s,i,k,m,t} + W_{s,k,t} \le SupplyAvail_{s,k,t} + W_{s,k,t-1}
$$

7.6 Route capacity constraint

[
\sum_k Flow_{i,j,k,m,t} \le Cap_{i,j,m,t}
]

7.7 Fleet constraint

[
\sum_{i,j,k} \frac{Flow_{i,j,k,m,t}}{Payload_m} \le VehiclesAvailable_{m,t}
]

7.8 Storage capacity constraint

[
\sum_k I_{i,k,t} \le StorageCap_i
]

7.9 Budget constraint

[
ProcCost_t + TransCost_t + HoldCost_t + ReserveCost_t \le Budget_t + G_t
]

If funding gaps are not allowed, set (G_t = 0).

7.10 Service level constraint

[
S_{i,k,t} \le MaxShortagePct_{i,k,t} \times D_{i,k,t}
]

This can be stricter for priority cities.

---

8. Objective function

The model should minimize total system cost plus shortage risk.

[
\min Z =
\sum_t \Big[
ProcurementCost_t

- TransportationCost_t
- HoldingCost_t
- ReserveHoldingCost_t
- RouteActivationCost_t
- ShortagePenalty_t

- InstabilityPenalty_t
- FundingGapPenalty_t

  \Big]

  ]

Where:

Procurement cost

Food purchase cost

Transportation cost

Movement by link and mode

Holding cost

Normal inventory carrying cost

Reserve holding cost

Cost of protecting strategic reserves

Shortage penalty

Very high penalty for unmet demand

Instability penalty

Penalty for excessive fluctuation in city availability from one period to the next

Example:

[
InstabilityPenalty_t = \sum_{i,k} \omega_{i,k} \cdot |AvailabilityRatio_{i,k,t} - AvailabilityRatio_{i,k,t-1}|
]

This is important because your goal is not just low cost, but also **stable food availability**.

---

9. Outputs the model should produce

The system should generate eight main outputs.

Output 1 — City requirement report

For each city and period:

- forecast demand
- served demand
- reserve target
- shortage
- days of cover

Output 2 — Distribution plan

For each route and mode:

- origin
- destination
- product
- quantity
- trips required
- cost

Output 3 — Transportation requirements

By period:

- tonnage to move
- ton-km
- trucks/railcars/vessels needed
- fleet utilization

Output 4 — Funding requirements

By period:

- procurement funds

- transport funds
- storage funds
- reserve funds
- total funds
- funding gap

Output 5 — Reserve sufficiency dashboard

By city:

- actual reserve
- target reserve
- gap
- days of cover

Output 6 — Shortage and vulnerability map

By city and product:

- shortage amount
- service level
- disruption sensitivity

- priority rank

Output 7 — Route criticality report

- most important links
- overloaded links
- most expensive links
- most fragile links

Output 8 — Scenario comparison

Compares baseline vs disrupted cases:

- extra funds needed
- extra reserve needed
- extra transport needed
- cities at highest risk

---

10. Graphic display structure

Here is a simple visual logic for a dashboard.

flowchart LR

A[Population & Consumption Data] --> B[Demand Forecast]

C[Supply / Import / Production] --> D[Supply Availability]

E[Routes / Modes / Capacities] --> F[Transport Network]

G[Budget / Cost / Funding] --> H[Financial Constraints]

I[Opening Inventory / Reserves] --> J[Inventory Logic]

B --> K[Optimization Engine]

D --> K

F --> K

H --> K

J --> K

K --> L[City Allocation Plan]

K --> M[Transport Requirement Plan]

K --> N[Reserve Targets & Gaps]

K --> O[Shortage Risk]

K --> P[Funding Requirement]

For a live decision dashboard, I would recommend these graphics:

- network map with route thickness by shipment volume
- heat map of shortage risk by city
- line chart of days-of-cover by city over time
- stacked cost chart by procurement, transport, storage, reserve
- scenario comparison chart baseline vs disruption

---

11. Model logic flow

Step 1

Load baseline data:
population, food norms, stocks, sources, links, budgets

Step 2

Forecast demand for each city and product

Step 3

Calculate required reserve by city and product

Step 4

Determine available supply from all sources

Step 5

Apply route, fleet, and storage constraints

Step 6

Optimize flows, inventory, and reserve decisions

Step 7

Calculate cost, funding requirement, and shortages

Step 8

Run scenarios for disruption, demand spikes, and budget cuts

---

12. Recommended model versions

Version 1 — Prototype

Best for quick proof of concept.

Use:

- 5 to 10 cities

- 3 to 5 products
- 12 monthly periods
- 2 modes
- fixed routes
- deterministic demand

Purpose:

prove the structure and produce first useful reports

Version 2 — Operational planning model

Use:

- 20 to 100 cities
- full food basket
- monthly or weekly periods
- reserve rules
- multiple supply nodes
- multiple transport modes
- budget constraints

Purpose:

planning and resource allocation

Version 3 — Risk and resilience model

Add:

- demand uncertainty
- supply shocks
- link disruption
- port closure
- fuel spikes
- dynamic rerouting
- stochastic scenarios

Purpose:

stress testing and emergency preparedness

---

13. Recommended technical build

For practical development with me, I suggest this stack:

Data layer

Excel or CSV tables at first

Processing layer

Python with pandas

Optimization layer

Pyomo or OR-Tools

Visualization layer

Power BI, Excel dashboard, or Plotly

Mapping

NetworkX plus geographic plotting, or GIS later

---

14. Workbook / database structure

A good file structure would be:

Input sheets

- Cities
- Products
- DemandDrivers

- SupplySources
- OpeningInventory
- Links
- Fleet
- Budget
- Policy
- Scenarios

Calculation sheets

- DemandForecast
- ReserveTargets
- AvailableSupply
- RouteCapacity
- CostParameters

Output sheets

- CityRequirements
- DistributionPlan

- TransportRequirements
- FundingRequirements
- ReserveGap
- ShortageRisk
- ScenarioSummary

---

15. Example key performance indicators

The model should calculate at least these KPIs:

- total demand
- total served demand
- total shortage
- percent service level
- average days of cover
- reserve gap by city
- total tonnage moved
- total ton-km

- fleet utilization rate
- average transport cost per ton
- total funding needed
- funding gap
- network bottleneck index
- volatility index of city food availability

---

16. Suggested prototype assumptions

To avoid early complexity, start with these assumptions:

- one month time steps
- no substitution between products
- no perishability beyond a fixed loss rate
- fixed transport cost per ton-km
- fixed route capacities
- reserves only at city level
- no return trips modeled separately

- one objective: minimize cost plus shortage penalty

That gives us a clean, solvable first model.

---

17. Immediate next deliverable

The most useful next step is to convert this blueprint into a **working model skeleton** with three concrete pieces:

Piece 1 — exact Excel input template

with all sheets and column names

Piece 2 — mathematical formulation

clean set notation, parameters, variables, constraints, and objective

Piece 3 — Python prototype structure

folder layout, data import logic, and solver model outline

Below is the shortest practical mathematical summary.

---

18. Compact mathematical formulation

Sets

- (I): cities
- (S): supply nodes
- (K): products
- (M): transport modes
- (T): time periods
- (A): network arcs

Parameters

- (D_{i,k,t}): demand
- (Q_{s,k,t}): supply availability
- (C^{p}_{s,k,t}): procurement cost
- (C^{t}_{a,m,t}): transport cost
- (Cap_{a,m,t}): route capacity
- (StorCap_i): storage capacity
- (ResTarget_{i,k,t}): reserve target
- (Budget_t): budget
- (Penalty_{i,k,t}): shortage penalty

Variables

- (x_{a,k,m,t}): shipment on arc
- (i_{n,k,t}): inventory
- (r_{i,k,t}): reserve
- (s_{i,k,t}): shortage
- (g_t): funding gap

Objective

Minimize:

[

\sum C^p x + \sum C^t x + \sum Hold \cdot i + \sum Reserve \cdot r + \sum Penalty \cdot s + \sum GapPenalty \cdot g

]

Subject to

- demand balance
- inventory balance
- reserve target
- supply availability
- route capacity

- fleet availability
- storage capacity
- budget constraint
- nonnegativity

---

19. Best way to proceed

I suggest we now build this in this order:

Stage A: Excel input template

Stage B: mathematical formulation sheet

Stage C: Python prototype

Stage D: dashboard and graphics

The strongest next step is for me to draft the **actual Excel workbook structure with sheet names, column headers, and sample rows** so the model becomes immediately buildable.