# Palantir Developer Technical Challenge

## 1. Introduction

The objective of this ontology design was to integrate siloed data across orders, inventory, warehouses and shipments into a unified data model. By creating standardized relationships, constraints and definitions, this ontology enables interoperability across business systems, supports automation and provides a foundation for advanced analytics such as demand forecasting and supply chain optimization.

## 2. Understanding and Analysing the Data

Before designing the ontology, an assessment of the available datasets was performed. The analysis highlighted several challenges:

* **Entity Identification**: The core entities (Customer, Order, Product, Inventory, Warehouse, Shipment, Carrier) were identified as the building blocks of logistics data.
* **Disconnects Across Systems**:
  + Order records could not always be tied to the **specific warehouse** from which a product was shipped.
  + Shipment data lacked consistent alignment with **order status**, leading to discrepancies (e.g., orders marked as "Delivered" while shipments remained "In Transit").
  + Inventory data was isolated, making it difficult to reconcile product movement against customer demand.
* **Impact of Data Silos**: Because data existed in separate systems, operational teams could not perform **cross-functional analysis** (e.g., shipment delays affecting inventory levels).

These observations shaped the ontology by ensuring that **all relationships explicitly capture the links between orders, warehouses, inventory, and shipments**.

## 3. Assumptions Made

To bridge gaps and design a usable ontology, several assumptions were necessary:

1. **Order Fulfillment**: Each order is fulfilled from a warehouse, even if this link was not always explicit in source data.
2. **Shipment Status Hierarchy**: Shipment status is considered the source of truth for delivery progress, with order status aligned accordingly.
3. **Inventory Tracking**: Inventory is warehouse-specific, and all stock/reservation updates are tied to product-warehouse combinations.
4. **Date Constraints:**
   1. Order dates cannot be in the future.
   2. Shipment dates must be on or after the order date.
   3. Estimated delivery dates must be greater than or equal to the shipment date.
   4. Actual delivery dates must be greater than or equal to the shipment date, ensuring logical delivery timelines.
5. **Carrier Service Levels**: Service levels (Standard, Express, Overnight) are standardized across carriers for consistency.

These assumptions allowed for the design of a **consistent and enforceable data model**.

## 4. Design Approach

The ontology was designed with the following guiding principles:

**a. Entity-Centric Modeling**

* Core entities capture business objects with well-defined attributes and constraints (e.g., stock must be non-negative, email must be valid).

**b. Relationship Definition**

* Orders → Shipments, Product
* Products → Inventory
* Customers → Orders
* Shipments → Warehouses & Carriers
* Inventory → Warehouses
* Warehouses → Carriers

**c. Business Rules and Constraints**

* Composite keys prevent duplication.
* Validation rules (e.g., future dates not allowed) ensure data quality.
* Enumerations standardize business states (Pending, Shipped, Delivered).

## 5. Business Context Considered

In building this ontology, I considered both the **current challenges** and the **company’s goals for process improvement**:

* The company aims to **streamline operations** by unifying data across logistics, inventory, and customer management.
* Improving **visibility of warehouse-level operations** is critical to preventing delays and shortages.
* Standardizing shipment and order tracking helps reduce **status discrepancies** that confuse both customers and staff.
* A scalable model supports future automation such as:
  + **Predictive restocking** using shipment delay patterns.
  + **Customer insights** through order histories.
  + **Carrier performance benchmarking** for service level agreement (SLA) compliance.

## 6. Cross-Functional Data Use

The ontology enables reuse of data beyond its original system:

* Shipment delays feed into **inventory forecasting**.
* Customer orders drive **demand prediction** and safety stock planning.
* Carrier delivery times support **vendor performance management**.

## 7. Benefits

* **Interoperability**: Breaks down silos by unifying order, shipment, and inventory data.
* **Automation**: Provides the foundation for automated restock recommendations, shipment alerts, and performance dashboards.
* **Operational Insights**: Enables predictive analytics and proactive decision-making.

## 8. Conclusion

By analysing existing disconnects, making informed assumptions, and considering the company’s process improvement goals, this ontology provides a robust foundation for supply chain automation. It enforces data quality, clarifies entity relationships, and unlocks new opportunities for operational efficiency within Palantir Foundry.

# Automation Logic Report

## 1. Objective

The goal of this automation logic is to **trigger restocking recommendations** whenever stock levels fall below a calculated threshold. This ensures warehouses maintain sufficient inventory to meet customer demand without overstocking.

## 2. Key Data Considered

The logic leverages three main inputs:

* **Sales Velocity (30 days)**: Units sold per day per product-warehouse, based on order and shipment data.
* **Average Shipment Times**: Average delivery duration from each warehouse to customers.
* **Current Inventory Levels**: Available stock after accounting for reserved quantities.

### 3. Automation Logic

1. **Calculate Sales Velocity**:
   * For each product-warehouse pair, determine daily sales rate over the last 30 days.
2. **Determine Safety Stock**:
   * Safety stock = sales velocity × 14 days (buffer against demand surges).
3. **Compute Reorder Point**:
   * Reorder point = sales velocity × (average shipment time + 7 days threshold).
   * If available stock ≤ reorder point, the system flags restocking.
4. **Calculate Recommended Restock Quantity**:
   * Target stock = safety stock + (sales velocity × 30 days replenishment period).
   * Recommended restock quantity = target stock − available stock.
5. **Prioritize by Urgency**:
   * Urgency score = (reorder point − available stock) ÷ reorder point × 100.
   * Recommendations are sorted by urgency for operational priority.

## 4. Output

The system produces a **list of restocking recommendations** with:

* product\_id
* warehouse\_id
* recommended\_restock\_quantity
* Additional insights (urgency score, sales velocity, shipment time).

Results are available as:

* **Console report** for quick review.
* **CSV export** for operational teams.
* **JSON export** for system-to-system integration.

## 5. Benefits

* **Prevents stockouts** by forecasting needs before shortages occur.
* **Improves efficiency** by aligning stock replenishment with shipment lead times.
* **Scalable** to additional products, warehouses, and carriers.

# Business Value Report

## 1. Business Problem

The company’s order, inventory, and shipment data were siloed across systems. This caused:

* **Order/Shipment Mismatches** → Unclear delivery progress.
* **Inventory Blind Spots** → No traceability of stock levels per warehouse.
* **Inefficient Decision-Making** → Heavy manual effort, delays, and missed opportunities.

## 2. The Solution

We built an **ontology-driven data model** and **automation logic** that:

* **Integrates Orders, Inventory, and Shipments** into one interoperable framework.
* **Applies Constraints & Standards** (e.g., shipment dates, carrier service levels).
* **Automates Inventory Restock Recommendations** using sales velocity, shipment lead times, and safety stock.
* **Generates Business Insights** such as:
  + Average delivery times per carrier.
  + Top 5 best-selling products last quarter.
  + Warehouse-level inventory shortage visualization.
  + (Advanced) Supplier integration to link procurement with demand.

## 3. Business Value Delivered

**Operational Efficiency**

* Eliminates manual reconciliation of disconnected datasets.
* Enables real-time visibility across the supply chain.
* **Reduces employee workload** by automating repetitive monitoring and reporting.

**Reduced Stockouts & Delays**

* Predictive restocking ensures customer demand is met.
* Shipment data feeds directly into inventory planning.
* **Shifts operations from reactive to predictive**, enabling proactive problem-solving.

**Scalable & Reusable Data Foundation**

* Ontology supports multiple business functions beyond logistics (e.g., finance, customer service, analytics).
* Provides a foundation for future automation (AI forecasting, supplier optimization).

**Improved Customer Experience**

* Faster, more reliable deliveries.
* Transparent tracking of orders and inventory.
* Better alignment of stock with **top-selling products**.

## 4. Strategic Impact

By breaking down silos and enforcing a **single source of truth**, the company:

* Gains **cross-functional insights** (e.g., using shipment delays to adjust safety stock).
* Aligns supply chain operations with business growth goals.
* Establishes a **scalable digital backbone** for automation and advanced analytics.