# 05 Supply Chain Data Integration System

# Supply Chain Data Integration System

## Project Overview

This project aims to develop a practical supply chain data integration system that consolidates data from two accessible sources. The solution will use Python for transformations like lead time calculations, order cycle analysis, and inventory metrics. The system will implement a well-designed star schema in BigQuery with specialized data marts for vendor management and inventory analysis.

## Data Sources and Available APIs

### Primary Data Sources

1. **Global Superstore Dataset**
   - Source: Global Superstore Dataset
   - Format: Excel file with multiple sheets (Orders, Returns, People)
   - Acquisition Method: Free download from Kaggle (requires Kaggle account)
   - Refresh Method: One-time download with simulated incremental updates
   - Key entities: Orders, products, customers, shipping, returns
   - Documentation: Data dictionary included in the dataset
   - Sample Access Code:

```python
import kaggle
import pandas as pd

# Download the dataset
kaggle.api.authenticate()
kaggle.api.dataset_download_files('rohitsahoo/sales-forecasting', path='./data', unzip=True)

# Load the data
orders = pd.read_excel('./data/Global_Superstore.xlsx', sheet_name='Orders')
returns = pd.read_excel('./data/Global_Superstore.xlsx', sheet_name='Returns')

# Sample data exploration
print(f"Total orders: {len(orders)}")
print(f"Date range: {orders['Order Date'].min()} to {orders['Order Date'].max()}")
print(f"Product categories: {orders['Category'].unique()}")
print(f"Return rate: {len(returns) / len(orders):.2%}")
```

2. **Inventory and Price Simulation API**
   - Source: Fake Store API + Custom Inventory Simulation
   - Format: REST API with JSON responses + Python simulation for inventory levels
   - Acquisition Method: Public API (no authentication) + custom Python code
   - Refresh Method: API calls for product data, simulated daily inventory updates
   - Key entities: Products, categories, inventory levels, price history
   - Sample Access Code:

```python
import requests
import pandas as pd
import random
import datetime

# Get products from Fake Store API
```

```python
def get_products():
    response = requests.get('https://fakestoreapi.com/products')
    return response.json()

# Simulate inventory levels for products
def simulate_inventory(products, days=30):
    inventory_history = []

    start_date = datetime.datetime.now() - datetime.timedelta(days=days)

    for product in products:
        # Initial inventory level
        base_inventory = random.randint(20, 200)
        daily_demand = random.randint(1, 10)

        for day in range(days):
            current_date = start_date + datetime.timedelta(days=day)

            # Simulate inventory reduction
            daily_sales = min(random.randint(0, daily_demand), base_inventory)
            base_inventory -= daily_sales

            # Simulate restocking every 7 days
            if day % 7 == 0:
                restock_amount = random.randint(30, 100)
                base_inventory += restock_amount

            inventory_history.append({
                'product_id': product['id'],
                'product_name': product['title'],
                'date': current_date.strftime('%Y-%m-%d'),
                'inventory_level': base_inventory,
                'category': product['category'],
                'price': product['price'],
                'restock': 1 if day % 7 == 0 else 0
            })

    return pd.DataFrame(inventory_history)

# Get products and simulate inventory
products = get_products()
inventory_df = simulate_inventory(products)
```

## Core Functional Requirements

1. **Data Extraction and Integration**
   - Build connectors for Global Superstore dataset and Fake Store API
   - Implement inventory simulation for daily stock levels
   - Create data validation checks for supply chain data integrity
   - Develop reconciliation processes between sources
   - Simulate incremental data loads for both sources
2. **Supply Chain Metrics Calculation**
   - Implement lead time calculations based on order and ship dates
   - Create order cycle time analysis from order to delivery
   - Develop inventory turnover and days-on-hand metrics
   - Calculate on-time delivery and order fill rates
   - Implement product category performance metrics
3. **Dimensional Modeling and Storage**

- Design and implement supply chain star schema in BigQuery
- Create dimensions for products, locations, customers, and time
- Implement fact tables for orders, shipments, and inventory
- Apply appropriate partitioning and clustering for supply chain analytics
- Implement aggregation tables for common supply chain metrics

4. **Data Marts and Analytics**
    - Create vendor/supplier performance data mart
    - Develop inventory analysis data mart with stock level metrics
    - Build order fulfillment analytics mart
    - Implement product category performance analytics
    - Create shipping and logistics analysis mart
5. **Monitoring and Visualization**
    - Develop Streamlit dashboard for supply chain KPIs
    - Create visualizations for inventory levels and movements
    - Implement product performance scorecards
    - Build geographic shipping performance visualization
    - Create alerting for inventory exceptions and anomalies

## Standard Functional Scope

### Phase 1: Foundation

- Initial supply chain data model design
- Global Superstore data loading and processing
- Fake Store API integration and inventory simulation
- Preliminary BigQuery warehouse setup
- Simple pipeline orchestration with error handling

### Phase 2: Enhancement

- Implement core supply chain metrics calculation
- Develop inventory analysis functionality
- Create order fulfillment analytics
- Build initial Streamlit dashboard for key metrics
- Implement data quality and reconciliation processes

### Phase 3: Optimization

- Add remaining data marts for comprehensive analytics
- Implement advanced metrics and KPIs
- Create interactive supply chain visualizations
- Develop exception monitoring and alerting
- Optimize BigQuery performance for supply chain analytics

## Definition of Done

### For Data Pipeline Components

- Global Superstore data is correctly loaded and processed
- Fake Store API connector successfully retrieves product data
- Inventory simulation generates realistic daily stock levels
- Data validation identifies inconsistencies across sources
- Transformations properly calculate all supply chain metrics
- Error handling successfully manages and logs issues
- Full documentation of supply chain data pipeline is complete

### For Data Warehouse

- Supply chain star schema follows dimensional modeling best practices
- Product and location dimensions are properly structured
- Fact tables correctly store metrics with appropriate granularity
- BigQuery partitioning and clustering optimize query performance
- Data marts accurately represent supply chain domains
- Query performance meets requirements for operational reporting

### For Analytics and Monitoring

- Streamlit dashboard accurately displays supply chain KPIs
- Inventory analytics provide accurate visibility to stock levels
- Order fulfillment metrics show realistic performance
- Geographic visualizations show shipping patterns correctly
- Exception alerts trigger appropriately for inventory anomalies
- All metrics and calculations are documented with business definitions