End-to-End Supply Chain Analytics API using FastAPI, Supabase, and Python

1. Introduction

Modern supply chains generate massive volumes of transactional and operational data, such as supplier shipments, customer sales orders, and warehouse inventory snapshots. Converting this raw data into actionable insights is essential for improving efficiency, forecasting demand, and mitigating risks such as stockouts or delayed shipments.

Traditional analytics approaches often require manual data processing, scattered spreadsheets, or heavy enterprise systems that are difficult to scale. To address these challenges, this project develops an automated, API-driven supply chain analytics microservice that streamlines the entire workflow.

The platform is built using FastAPI for lightweight yet high-performance REST APIs and integrates with Supabase (PostgreSQL backend) for persistent data storage. For prototyping or quick testing, the system can also run entirely in-memory, making it flexible for both enterprise-scale deployments and academic or proof-of-concept use cases.

Data Ingestion: Seamlessly ingest structured data from suppliers, sales orders, and inventory systems via JSON payloads or CSV uploads.

Forecasting: Provides multiple demand forecasting methods including Seasonal Moving Average (SMA), Linear Regression, and an Ensemble approach combining both.

API-First Design: Exposes analytics results via REST endpoints for integration into dashboards, BI tools, or downstream decision support systems.

With this design, the system acts as a scalable backend service for supply chain analytics, enabling real-time monitoring and forecasting of complex supply chain operations.

2. Objectives

- Automate ingestion of supply chain data in JSON/CSV format.
- Provide short-term demand forecasting using SMA, Linear Regression, and Ensemble methods.
- Make analytics accessible via REST API endpoints.
- Support both in-memory execution and persistent database storage (Supabase).

3. Methodology

- This project follows a modular API-driven analytics workflow, combining data ingestion and forecasting into a single, lightweight service.

3.1 Architecture

The system is designed around FastAPI for performance and modularity, with optional database integration for scalability.

- **FastAPI** → High-performance API framework for exposing ingestion and forecast endpoints.
- **Supabase (PostgreSQL)** → Cloud-hosted database for persistent storage and production deployment (optional).
- **Pandas & NumPy** → Core libraries for data wrangling, aggregation, and forecasting.
- **SciPy & scikit-learn** → Used for statistical regression and machine learning forecasting methods.
- **dotenv** → Secure handling of environment variables (e.g., API keys, database credentials). API using FastAPI, Supabase, and Python.

3.2 Data Model (Logical Tables)

Data is structured into logical tables to mirror key entities in a supply chain:

- **suppliers** → Supplier details (ID, metadata).
- **skus** → Stock-keeping units (unique product identifiers).
- **supplier_shipments** → Shipment records (supplier, SKU, quantity, lead time, cost).
- **sales_orders** → Customer orders (SKU, date, quantity, price).
- **inventory_snapshots** → Time-based inventory levels (on-hand, in-transit).
- **kpi_daily** → Pre-computed KPIs (future scope, not implemented in current version).
- **forecasts** → Forecast results (SKU, horizon, predicted demand).

3.3 Forecasting Methods

1. **Seasonal Moving Average (SMA)**
   Uses a rolling average over a specified season (e.g., 7 days) to smooth demand fluctuations and predict future quantities.
2. **Linear Regression**
   Fits a regression line over historical daily demand trends to extrapolate future demand.
3. **Ensemble Method**
   Combines SMA and Linear Regression predictions with weighted averaging (60% SMA, 40% Linear) for more robust results.

4. **Fallback Simple Forecast**
   If insufficient data is available, the model defaults to using the mean demand as a simple forecast.

3.4 API Workflow

1. **Data Ingestion (/ingest)**
   - Accepts data via JSON or CSV uploads.
   - Stores in Supabase (if configured) or in-memory tables.
2. **Model Training (/retrain)**
   - Trains forecasting models (SMA, Linear, Ensemble) for all SKUs.
   - Generates forecasts for a specified horizon (default: 14 days).
3. **Forecast Access (/forecast/{sku})**
   - Returns SKU-level forecasts for the requested horizon.

4. Results

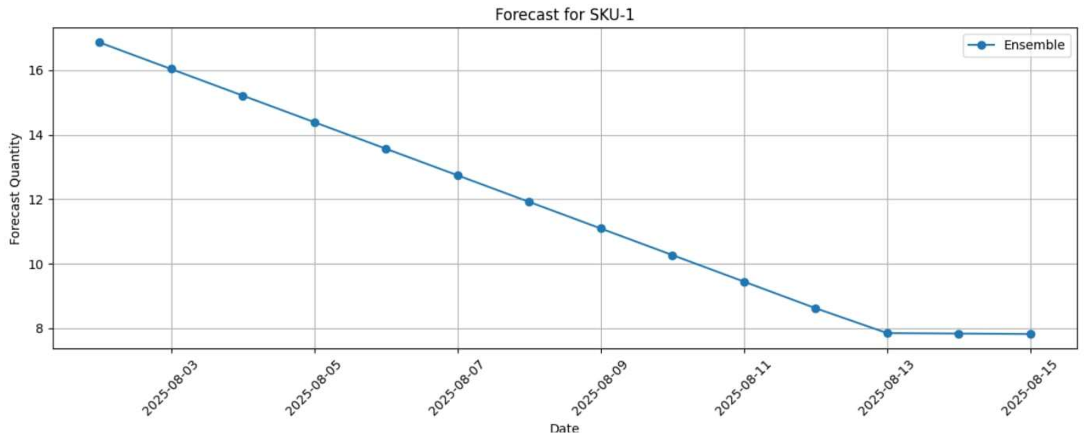
Example Input Data (Sales Orders)

Order ID	SKU	Quantity	Order Date
ORD-0001	SKU-1	120	2025-07-02
ORD-0002	SKU-1	140	2025-07-09
ORD-0003	SKU-2	80	2025-07-03

Example Forecasts (Ensemble Method)

SKU	Forecast Date	Horizon (Days)	Forecast Qty	Method
SKU-1	**2025-08-02**	**1**	**16.86**	**Ensemble**
SKU-1	**2025-08-03**	**2**	**16.03**	**Ensemble**
SKU-1	**2025-08-04**	**3**	**15.21**	**Ensemble**
SKU-1	**2025-08-05**	**4**	**14.39**	**Ensemble**
SKU-1	**2025-08-06**	**5**	**13.56**	**Ensemble**

```
Forecast sample:
     sku        forecast_date  horizon_days  forecast_qty    method
0  SKU-1  2025-08-02T00:00:00             1     16.862984  Ensemble
1  SKU-1  2025-08-03T00:00:00             2     16.038705  Ensemble
2  SKU-1  2025-08-04T00:00:00             3     15.214426  Ensemble
3  SKU-1  2025-08-05T00:00:00             4     14.390148  Ensemble
4  SKU-1  2025-08-06T00:00:00             5     13.565869  Ensemble
```



Insights from Forecast Plot

- The forecast for SKU-1 shows a gradual decline in demand from ~16.8 units to ~7.8 units over the forecast horizon.
- The Ensemble method balances seasonal patterns (SMA) with long-term trends (Linear Regression), resulting in smoother predictions.
- This insight can help supply chain managers reduce overstocking by aligning procurement with expected declining demand.