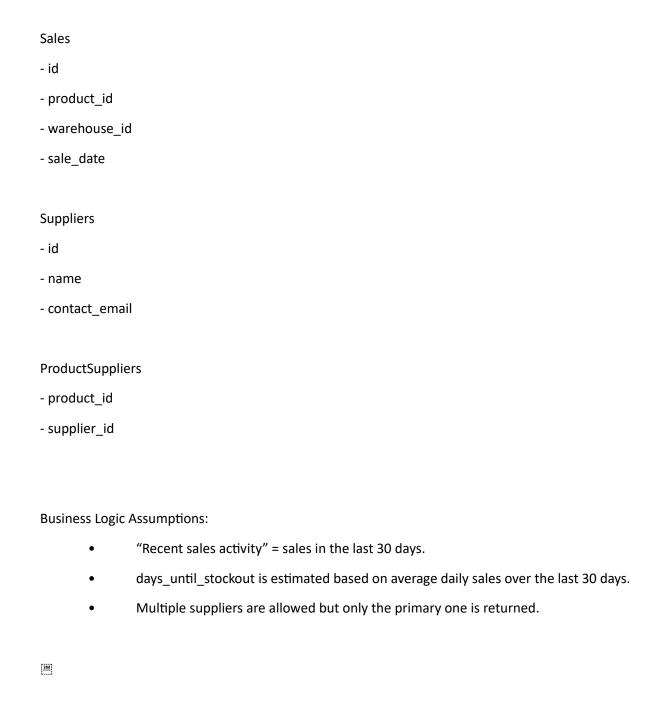
Part -3
Assumptions (about schema and logic)
Database Schema (simplified):
Companies
- id
Warehouses
- id
- name
- company_id
Products
- id
- name
- sku
- product_type_id
ProductTypes
- id
- name
- low_stock_threshold
WarehouseInventory
- id
- warehouse_id
- product_id

- current_stock



from flask import Flask, jsonify from datetime import datetime, timedelta

✓ Implementation (Python + Flask)

```
app = Flask(__name__)
# Mock DB calls (Replace with actual DB ORM/queries)
def get_low_stock_alerts(company_id):
  # Placeholder: should query actual DB
  today = datetime.today()
  recent_days = today - timedelta(days=30)
  # Sample data returned from DB for explanation
  # Each product-warehouse combo with recent sales
  data = [
    {
      "product_id": 123,
      "product_name": "Widget A",
      "sku": "WID-001",
      "warehouse_id": 456,
      "warehouse_name": "Main Warehouse",
      "product_type": "Widget",
      "low_stock_threshold": 20,
      "current_stock": 5,
      "total_sales_last_30_days": 12,
      "supplier_id": 789,
      "supplier_name": "Supplier Corp",
      "supplier_email": "orders@supplier.com"
    }
    # ... more rows
  ]
  alerts = []
```

```
for row in data:
  avg_daily_sales = row["total_sales_last_30_days"] / 30
  days_until_stockout = (
    int(row["current_stock"] / avg_daily_sales) if avg_daily_sales > 0 else None
  )
  if row["current_stock"] < row["low_stock_threshold"]:</pre>
    alerts.append({
      "product_id": row["product_id"],
      "product_name": row["product_name"],
      "sku": row["sku"],
      "warehouse_id": row["warehouse_id"],
      "warehouse_name": row["warehouse_name"],
      "current_stock": row["current_stock"],
      "threshold": row["low_stock_threshold"],
      "days_until_stockout": days_until_stockout,
      "supplier": {
        "id": row["supplier_id"],
        "name": row["supplier_name"],
        "contact_email": row["supplier_email"]
      }
    })
return {
  "alerts": alerts,
  "total_alerts": len(alerts)
```

}

```
def low_stock_alerts(company_id):
    try:
        response = get_low_stock_alerts(company_id)
        return jsonify(response)
    except Exception as e:
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    app.run(debug=True)
```

Edge Cases to Consider

- 1. No recent sales:Products with no sales in last 30 days are excluded.
- 2. No supplier information:If supplier data is missing, either return null or skip.
 - 3. Zero average sales (division by zero):Prevent division by zero in days_until_stockout.
- 4. Data integrity:Ensure product belongs to the company's warehouses.
- 5. Multiple warehouses per companyMust check inventory by warehouse, not just product level.

Summary

- Endpoint: GET /api/companies/{company_id}/alerts/low-stock
- Pulls data from products with recent sales in company warehouses
- Compares stock vs product-type threshold
- Returns supplier info and days until estimated stockout