Part one: Debug

Issues Identified

1. Error Handling

- **Problem**: No validation for missing fields
- **Impact**: Server crashes with KeyError

2. Misplaced Entities

- **Problem**: Product has warehouse_id (should be in Inventory only)
- **Impact**: Violates "products in multiple warehouses" requirement

3. Rollback

- **Problem**: No exception in case of error or mid-termination.
- **Impact**: Lead to corrupted data.

Improved code!

```
@app.route('/api/products', methods=['POST'])
def create_product():
    try:

        # Check if the server is connected.
        db.session.execute('SELECT 1')

        data = request.json

        if not data:
            return {"error": "Invalid input"}

        # Check for fields
        fields = ['name', 'sku', 'price', 'warehouse_id', 'initial_quantity']

        for field in fields:
            if field not in data:
                return {"error": f"Missing field: {field}"}
        # Unique SKU check
        existing_sku = product.query.filter_by(sku=data['sku']).first()
        if existing_sku:
            return {"error": "SKU already exists"}


        db.session.begin()


        # Create new product
        product = Product(
            name=data['name'],
            sku=data['sku'],
            price=data['price']
            # Not need for warhouse_id here.
            # Assuming inventory Entity has warehouse_id key, and it is not present in Product
Entity.
        )

        db.session.add(product)
        db.session.flush() # not commiting here


        # Update inventory count
        inventory = Inventory(
            product_id=product.id,
            warehouse_id=data['warehouse_id'],
            quantity=data['initial_quantity']
        )

        db.session.add(inventory)
        db.session.commit()

        return {"message": "Product created", "product_id": product.id}

    except SQLAlchemyError:
        db.session.rollback()
        return {"error" : "Database not connected!"}

    except Exception as e:
        db.session.rollback()
        return {"error": "Internal Error!"}
```

Questions for team:

- Need database schema for complete analysis

Part 2: SQL

Questions for Team:

- Are Products in bundles Reservered or gonna pick from the stock!

```
CREATE TABLE Companies (
   CompanyID INT PRIMARY KEY,
   CompanyName VARCHAR(255) NOT NULL,
   Location VARCHAR(255) NOT NULL
);

CREATE TABLE Suppliers(
   SupplierID INT PRIMARY KEY,
   SupplierName VARCHAR(255) NOT NULL,
   Contact_email VARCHAR(255) NOT NULL
);

CREATE TABLE Categories (
   CategoryID INT PRIMARY KEY,
   CategoryName VARCHAR(255) NOT NULL,
   min_Quantity INT NOT NULL DEFAULT 5
);

CREATE TABLE Warehouse(
   WarehouseID INT PRIMARY KEY,
   WarehouseName VARCHAR(255) NOT NULL,
   Location VARCHAR(255) NOT NULL,
   CompanyID INT,
   FOREIGN KEY (CompanyID) REFERENCES Companies(CompanyID)
);

CREATE TABLE Product (
   ProductID INT PRIMARY KEY,
   SKU_ID VARCHAR(50) NOT NULL UNIQUE,
   ProductName VARCHAR(255) NOT NULL,
   Price DECIMAL(10, 2) NOT NULL,
   SupplierID INT,
   CategoryID INT,
   days_until_restock INT,
   is_Bundle BOOLEAN NOT NULL DEFAULT FALSE,
   FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID),
   FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

CREATE TABLE BundleItems (
   BundleID INT,
   ProductID INT,
   Quantity INT DEFAULT 1,
   PRIMARY KEY (BundleID, ProductID),
   FOREIGN KEY (BundleID) REFERENCES Product(ProductID),
   FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE Inventory (
   InventoryID INT PRIMARY KEY,
   ProductID INT NOT NULL,
   Quantity INT NOT NULL,
   WarehouseID INT NOT NULL,
   last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   last_stock_added INT DEFAULT 0,
   is_active BOOLEAN NOT NULL DEFAULT TRUE,
   FOREIGN KEY (WarehouseID) REFERENCES Warehouse(WarehouseID),
   FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);

CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   OrderDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   CustomerID INT,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE OrderItems (
   OrderItemID INT PRIMARY KEY,
   OrderID INT NOT NULL,
   ProductID INT NOT NULL,
   Quantity INT NOT NULL,
   FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
   FOREIGN KEY (ProductID) REFERENCES Product(ProductID)
);
```

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(255) NOT NULL,
    ContactEmail VARCHAR(255) NOT NULL
);
```

Part 3:

Assumptions

- Used above (part 2) Database Schema as reference

```
@app.route('/api/companies/<int:company_id>/alerts/low-stock', methods=['GET'])
def get_low_stock_alerts(company_id):
    try:
        # check for company
        company = Company.query.get(company_id)
        if not company:
            return {"error": "Company not found"}

        # Fetch from DB
        alert_query = db.session.query(Product.ProductID.label("ProductID"),
                                Product.ProductName.label("ProductName"),
                                Product.SKU_ID.label("SKU_ID"),
                                Inventory.InventoryID.label("InventoryID"),
                                Inventory.Quantity.label("Quantity"),
                                Categories.min_Quantity.label("min_Quantity"),
                                Inventory.days_until_restock.label("days_until_restock"),
                                Inventory.is_active.label("is_active"),
                                Warehouse.WarehouseName.label("WarehouseName"),
                                Warehouse.WarehouseID.label("WarehouseID"),
                                Supplier.SupplierID.label("SupplierID"),
                                Supplier.SupplierName.label("SupplierName"),
                                Supplier.Contact_email.label("Contact_email")

        ).join(
            Inventory, Product.ProductID == Inventory.ProductID # Join invertory with reference to
productID's
        ).join(
            Warehouse, Inventory.WarehouseID == Warehouse.WarehouseID # Join warehouse
with reference to WarehouseID's
        ).join(
            Supplier, Product.SupplierID == Supplier.SupplierID # Join supplier with reference to
SupplierID's
        ).join(
            Categories, Product.CategoryID == Categories.CategoryID # Join categories with
reference to CategoryID's
        ).filter(
            # Check for quantity
            Warehouse.CompanyID == company_id,
            Inventory.is_active == True,  # Only active inventory
            Inventory.Quantity <= Categories.min_Quantity
        ).all()

        alerts = [] # Notifications

        """Expected Response Format:
{
  "alerts": [
    {
      "product_id": 123,
      "product_name": "Widget A",
      "sku": "WID-001",
      "warehouse_id": 456,
      "warehouse_name": "Main Warehouse",
      "current_stock": 5,
      "threshold": 20,
      "days_until_stockout": 12,
      "supplier": {
        "id": 789,
        "name": "Supplier Corp",
        "contact_email": "orders@supplier.com"
      }
    }
  ],
  "total_alerts": 1
}
"""

        for alert in alert_query:
            notification = {
                "product_id": alert.ProductID,
                "product_name": alert.ProductName,
                "sku": alert.SKU_ID,
                "warehouse_id": alert.WarehouseID,
                "warehouse_name": alert.WarehouseName,
                "current_stock": alert.Quantity,
                "threshold": alert.min_Quantity,
                "days_until_restock": alert.days_until_restock,
                "supplier":{
                    "id": alert.SupplierID,
                    "name": alert.SupplierName,
                    "contact_email": alert.Contact_email
                }
            }
            alerts.append(notification)
        return {
            "alerts": alerts,
            "total_alerts": len(alerts)
        }
    except Exception as e:
        return {"error": "Internal Error"}
```