## ISSUES WITH THE CODE

1. **No validation of request JSON**
   - Code assumes request.json is always present and valid.
2. **No validation for required fields**
   - Missing fields like name, sku, or price will cause runtime errors.
3. **SKU uniqueness is not enforced**
   - Duplicate SKUs can be inserted, violating business rules.
4. **Product is wrongly linked to a single warehouse**
   - Products should be independent of warehouses.
5. **Improper database design**
   - Warehouse ID should be stored in inventory, not in product.
6. **Price is stored without decimal safety**
   - Using float for price can cause precision errors.
7. **Multiple database commits**
   - Product and inventory are committed separately.
8. **No transaction handling**
   - If inventory insert fails, product remains saved.
9. **Partial data persistence risk**
   - Database can contain products without inventory.
10. **No error handling**
    - Database or logic errors will crash the application.
11. **No handling of duplicate SKU errors**
    - Integrity errors are not caught.
12. A**ssumes initial_quantity is always present**
    - Missing field causes application crash.
13. **No rollback mechanism**
    - Failed operations do not revert previous changes.
14. **Risk of race conditions**
    - Concurrent requests can insert duplicate data.
15. **No HTTP status codes**

- Always returns success even on failure.

15 **No handling of optional fields**

- Optional fields are treated as mandatory.

16. **Tight coupling between product and inventory**

- Violates separation of concerns.

17.**No protection against invalid data types**

- Non-numeric price or quantity can be inserted.

## ***Corrected Code***

```
@app.route('/api/companies/<int:company_id>/alerts/low-stock', methods=['GET'])
def low_stock_alerts(company_id):
    """
    API to return low-stock alerts for a given company.
    It checks all warehouses of the company and finds products
    whose inventory is below the defined threshold.
    """

    # List to store all low-stock alerts
    alerts = []

    # -------------------------------------------------------
    # Step 1: Fetch all inventory records for the given company
    # -------------------------------------------------------
    inventories = (
        db.session.query(Inventory)
        .join(Warehouse)          # Join Inventory with Warehouse
        .join(Product)            # Join Inventory with Product
        .filter(Warehouse.company_id == company_id)
        .all()
    )
```

```
# ---------------------------------------------------------
# Step 2: Iterate through each inventory record
# ---------------------------------------------------------
for inventory in inventories:

    product = inventory.product      # Related product
    warehouse = inventory.warehouse  # Related warehouse

    # -----------------------------------------------------
    # Step 3: Skip products with no recent sales
    # (avg_daily_sales <= 0 means no recent activity)
    # -----------------------------------------------------
    if product.avg_daily_sales <= 0:
        continue

    # -----------------------------------------------------
    # Step 4: Check low-stock condition
    # If current stock is greater than or equal to threshold,
    # no alert is needed
    # -----------------------------------------------------
    if inventory.quantity >= product.low_stock_threshold:
        continue

    # -----------------------------------------------------
    # Step 5: Calculate days until stock runs out
    # Prevent division by zero (already ensured above)
    # -----------------------------------------------------
    days_until_stockout = int(
        inventory.quantity / product.avg_daily_sales
    )
    # -----------------------------------------------------

    # Step 6: Fetch supplier information (if available)

    # Assumption: one primary supplier per product

    # -----------------------------------------------------

    supplier = product.suppliers[0] if product.suppliers else None


    # -----------------------------------------------------

    # Step 7: Prepare alert response object

    # -----------------------------------------------------

    alert = {

        "product_id": product.id,

        "product_name": product.name,

        "sku": product.sku,

        "warehouse_id": warehouse.id,

        "warehouse_name": warehouse.name,

        "current_stock": inventory.quantity,

        "threshold": product.low_stock_threshold,

        "days_until_stockout": days_until_stockout,

        "supplier": {

            "id": supplier.id,

            "name": supplier.name,

            "contact_email": supplier.contact_email

        } if supplier else None

    }


    # Add alert to alerts list

    alerts.append(alert)


# ---------------------------------------------------------

# Step 8: Return final response
# ----------------------------------------------------------
return {
    "alerts": alerts,
    "total_alerts": len(alerts)
}, 200
```