Andy's Design Choice and Business logic for Order Manager Database Project

1. the primary key for Product table is the SKU only, which implies that there can only be 1 price for 1 SKU.

2. the description of SKU should have unique constraint. If two products have the same description, then why not assign them the same SKU?

3. the OrderDate in Orders table is generated by the time the new Orders_ID gensym is generated, even though no OrderRecord that is associated with this Orders_ID has been created. The shipping date for that Orders_ID is default to null.
Most important field in Order is the amount of OrderRecord that this Order contains.

4. There is a status Boolean for OrderRecord. If there is enough
Inventory for this SKU in Inventory, then the status will be changed to true and the quantity is deducted from the stock
Automatically. However, that order is not shipped unless this is the last record that changed from false to true. In other words,
One order is shipped when all the order record has enough stock

5. Here is how the shipDate is set using cascading triggers, one trigger another, until the date is set:
a. Insert an Order Record
b. after the insert, the "InsertUnitPriceInOrderRecord" trigger will auto select the price from inventory table and filled in the default guard value $ -1.00 .
c. If there is not enough stock, Boolean status remain false,
"InsufficientInventory" will call "InsufficientItems" procedures,
Although this record is still added for back ordering, an error message will be printed out.
d. If there is enough stock, status will be changed to true by the "ChangeOrderRecordStatus" trigger. This status change trigger will trigger two other triggers: First is "AutoDeductInventory",
Which will deduct the stock based on the UnitAmounts in this record. Second is "AutoShipDate". I implemented AutoShipDate using either Pure trigger (refer to

AutoShipDate2) or a trigger that is invoking a stored procedure, which links to external java program using prepared statement. Please refer to "AutoShipCheck" procedure which links to StoredProcedure.AutoShipCheck java method. Both methods will compare how many records in the records table has status true for one Order_ID which is one field of the newly inserted Record. If the number equals to the Record_Amounts field in Orders table, then every record is fulfilled and ShipDate is determined right now.

Other corner cases in this business model.

6. "ReturnItemsToInventory"
If one OrderRecord is true, which means the quantity of SKU is deducted, but this OrderRecord is not shipped as the
Orders_ID it belongs to has ShipDate = null. In this case, if we delete this Record, Then the SKU items are returned to Inventory. For all shipped records, we do not accept return and can't return those to Inventory.

7. "CheckRecordAmount" Trigger:
Since we have to specify Orders_ID in OrderRecord table, it's crucial to check that if this OrderRecord is invalid. If one order only has 2 records, then 3rd one is Considered as illegal.

8. "NoRecordForOrder"Trigger:
If one Record is the last existing Record for some Order_ID, then deleting this record will remove the Order_ID too.