**//apex trigger OrderTrigger**

trigger OrderTrigger on HandsMen\_Order\_\_c (before insert, before update) {

if (Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)) {

OrderTriggerHandler.validateOrderQuantity(Trigger.new);

}

}

**//apex class OrderTriggerHandler**

public class OrderTriggerHandler {

public static void validateOrderQuantity(List<HandsMen\_Order\_\_c> orderList) {

for (HandsMen\_Order\_\_c order : orderList) {

if (order.Status\_\_c == 'Confirmed') {

if (order.Quantity\_\_c == null || order.Quantity\_\_c <= 500) {

order.Quantity\_\_c.addError('For Status "Confirmed", Quantity must be more than 500.');

}

} else if (order.Status\_\_c == 'Pending') {

if (order.Quantity\_\_c == null || order.Quantity\_\_c <= 200) {

order.Quantity\_\_c.addError('For Status "Pending", Quantity must be more than 200.');

}

} else if (order.Status\_\_c == 'Rejection') {

if (order.Quantity\_\_c == null || order.Quantity\_\_c != 0) {

order.Quantity\_\_c.addError('For Status "Rejection", Quantity must be 0.');

}

}

}

System.debug('All records validated successfully.');

}

}

**// trigger OrderTotalTrigger**

trigger OrderTotalTrigger on HandsMen\_Order\_\_c (before insert, before update) {

Set<Id> productIds = new Set<Id>();

for (HandsMen\_Order\_\_c order : Trigger.new) {

if (order.HandsMen\_Product\_\_c != null) {

productIds.add(order.HandsMen\_Product\_\_c);

}

}

Map<Id, HandsMen\_Product\_\_c> productMap = new Map<Id, HandsMen\_Product\_\_c>(

[SELECT Id, Price\_\_c FROM HandsMen\_Product\_\_c WHERE Id IN :productIds]

);

for (HandsMen\_Order\_\_c order : Trigger.new) {

if (order.HandsMen\_Product\_\_c != null && productMap.containsKey(order.HandsMen\_Product\_\_c)) {

HandsMen\_Product\_\_c product = productMap.get(order.HandsMen\_Product\_\_c);

if (order.Quantity\_\_c != null) {

order.Total\_Amount\_\_c = order.Quantity\_\_c \* product.Price\_\_c;

}

}

}

}

**// trigger StockDeductionTrigger**

trigger StockDeductionTrigger on HandsMen\_Order\_\_c (after insert, after update) {

Set<Id> productIds = new Set<Id>();

for (HandsMen\_Order\_\_c order : Trigger.new) {

if (order.Status\_\_c == 'Confirmed' && order.HandsMen\_Product\_\_c != null) {

productIds.add(order.HandsMen\_Product\_\_c);

}

}

if (productIds.isEmpty()) return;

// Query related inventories based on product

Map<Id, Inventory\_\_c> inventoryMap = new Map<Id, Inventory\_\_c>(

[SELECT Id, Stock\_Quantity\_\_c, Product\_\_c

FROM Inventory\_\_c

WHERE Product\_\_c IN :productIds]

);

List<Inventory\_\_c> inventoriesToUpdate = new List<Inventory\_\_c>();

for (HandsMen\_Order\_\_c order : Trigger.new) {

if (order.Status\_\_c == 'Confirmed' && order.HandsMen\_Product\_\_c != null) {

for (Inventory\_\_c inv : inventoryMap.values()) {

if (inv.Product\_\_c == order.HandsMen\_Product\_\_c) {

inv.Stock\_Quantity\_\_c -= order.Quantity\_\_c;

inventoriesToUpdate.add(inv);

break;

}

}

}

}

if (!inventoriesToUpdate.isEmpty()) {

update inventoriesToUpdate;

} }

**//InventoryBatchJob**

global class InventoryBatchJob implements Database.Batchable<SObject>, Schedulable {

global Database.QueryLocator start(Database.BatchableContext BC) {

return Database.getQueryLocator(

'SELECT Id, Stock\_Quantity\_\_c FROM Product\_\_c WHERE Stock\_Quantity\_\_c < 10'

); }

global void execute(Database.BatchableContext BC, List<SObject> records) {

List<HandsMen\_Product\_\_c> productsToUpdate = new List<HandsMen\_Product\_\_c>();

// Cast SObject list to Product\_\_c list

for (SObject record : records) {

HandsMen\_Product\_\_c product = (HandsMen\_Product\_\_c) record;

product.Stock\_Quantity\_\_c += 50; // Restock logic

productsToUpdate.add(product);

}

if (!productsToUpdate.isEmpty()) {

try {

update productsToUpdate;

} catch (DmlException e) {

System.debug('Error updating inventory: ' + e.getMessage());

} } }

global void finish(Database.BatchableContext BC) {

System.debug('Inventory Sync Completed');

}

// Scheduler Method

global void execute(SchedulableContext SC) {

InventoryBatchJob batchJob = new InventoryBatchJob();

Database.executeBatch(batchJob, 200);

} }

**/ /Open Execute Anonymous Window**

System.schedule(‘Daily Inventory Sync’ , ‘0 0 0 \* \* ?’ , new InventoryBatchJob());