COMPSCI 310 Project Final Report*

Shaoxiang Qu[1] and Qixuan Yuan[2]

[1] Duke Kunshan University, Suzhou 215316, China
shaoxiang.qu@dukekunshan.edu.cn
[2] Duke Kunshan University, Suzhou 215316, China
qixuan.yuan@dukekunshan.edu.cn

Abstract. This report presents the complete design, implementation, and evaluation of a Multi-Warehouse Management and Fulfillment System developed using MySQL 8.0. The system addresses critical challenges in modern supply chain management including inventory inaccuracy, operational inefficiencies, and data fragmentation through a comprehensive database architecture. The implementation includes 10 core tables , 8 stored procedures, 4 MySQL functions, 11 database views, and 9 triggers that collectively enforce business rules at the database level. A Python-based command-line interface provides user access to 28 distinct operational functions covering warehouse management, inventory tracking, order processing, and analytical reporting. The system demonstrates robust ACID (Atomicity, Consistency, Isolation, Durability) compliance through InnoDB transactional support. This project establishes a production-ready foundation that balances sophistication with pragmatism, offering enterprises a cost-effective alternative to commercial warehouse management systems.

Keywords: Warehouse Management · MySQL · Database Design.

1 INTRODUCTION

The rapid expansion of global supply chains and e-commerce has created unprecedented demands for efficient warehouse management systems. In today's data-driven logistics environment, the effective utilization of big data has become a critical factor for supply chain competitiveness. As noted by Brinch (2018)[1], the value of big data in supply chain management manifests through three distinct dimensions: value discovery, value creation, and value capture. This framework provides essential guidance for organizations seeking to conceptualize strategies for leveraging data assets throughout their warehouse operations.

The integration of Industry 4.0 technologies, particularly the Internet of Things (IoT), represents a fundamental shift in warehouse management paradigms. Perera et al. (2023) [2] comprehensively demonstrate how IoT technologies—including RFID, QR codes, scanners, and advanced Warehouse Management Systems—transform warehouse operations by providing real-time data, enabling predictive maintenance, and improving inventory accuracy. Their systematic review highlights

* https://github.com/yqxcz06/Warehouse-Management-Database

that IoT integration substantially enhances warehouse management performance through increased operational efficiency, reduced costs, and improved customer satisfaction.

Academic research on warehouse operations has established comprehensive frameworks for understanding operational challenges and opportunities. Gu, Goetschalckx, and McGinnis (2006) [3] provide a seminal review that classifies warehouse operation problems according to fundamental functions: receiving, storage, order picking, and shipping. Their work bridges academic research and warehouse practice, explaining available planning models while identifying future research opportunities in operational optimization.

Commercial warehouse management solutions such as SAP Extended Warehouse Management and Oracle WMS Cloud offer sophisticated functionality but often present significant barriers including high licensing costs, complex implementation cycles, and vendor lock-in challenges. Conversely, open-source alternatives typically lack the granularity required for complex multi-warehouse operations, creating a middle ground that remains inadequately addressed.

This project addresses these gaps by developing a comprehensive MySQL-based warehouse management system that provides:

1. Enterprise-grade data integrity through database-level constraints and ACID compliance
2. Real-time operational capabilities
3. Advanced analytical functions via stored procedures and optimized views
4. Cost-effective deployment using entirely open-source technologies
5. Scalable architecture supporting growth from single-site to multi-regional operations

The system represents a practical implementation of database theory principles applied to real-world logistics challenges, demonstrating that thoughtful schema design combined with MySQL's mature feature set can deliver production-ready warehouse management capabilities while addressing the value dimensions identified by Brinch (2020) and leveraging the technological insights from Perera et al. (2023).

2 PROBLEM DEFINITION

2.1 Core Challenges

Modern warehouse management systems must address critical inefficiencies:

1. Inventory Inaccuracy: Physical stock often differs from system records
2. Operational Delays: Manual processes slow down workflows
3. Poor Space Utilization: Inefficient storage allocation
4. Data Fragmentation: Information silos across locations
5. Limited Scalability: Systems cannot handle business growth

2.2 Formal Requirements

Define WMS=(W, P, I, O) where:

- W: Warehouses with capacity constraints
- P: Products with attributes (volume, value, weight)
- I: Inventory states (total, available,locked)
- O: Orders (inbound / outbound)

Key Invariants

1. Inventory Conservation: $I_{\text{total}}(t) = I_{\text{total}}(t_0) + O_{\text{in}} - O_{\text{out}}$
2. Capacity Constraints: $\forall l \in \text{locations}: \quad \text{utilization}(l) \leq \text{capacity}(l)$
3. Availability Guarantee:$\forall o \in \text{orders}: \quad I_{\text{available}} \geq \text{required\_quantity}(O)$

2.3 Specific Problems Addressed

Real-time Inventory Tracking

Problem: Maintaining accurate inventory counts across multiple locations. Ensuring inventory accuracy across warehouses and storage locations. As it would be challenging for the management team with data lag and update conflicts.
Solution: Centralized inventory database with atomic updates:
Unified Inventory Table: inventory table tracks total, available, and locked quantities in real-time
Atomic Transactions: Ensure integrity and consistency of inventory updates
Row-level Locking: Prevent concurrent operation conflicts
Automatic Triggers: Update location utilization in real-time

Multi-warehouse Coordination

Problem: Optimizing inventory allocation across distributed warehouse network.
Solution: Unified database architecture with transfer mechanisms:
Global Inventory View: Supports cross-warehouse queries and transfers
Transfer Order Processing: Handles inventory movements between warehouses
Concurrency Control: Ensures data consistency during transfers

Expiration Management

Problem: Implementing First-In-First-Out (FIFO) or First-Expired-First-Out (FEFO) strategies for perishables, reducing expiration losses.
Solution: Batch-level tracking with date fields:
Batch Identification: Records production and expiration dates
Smart Picking: Queries sorted by expiration date
Early Warning System: Identifies soon-to-expire products

Automatic Reporting: Generates regular expiration risk assessments

Capacity Control

Problem: Preventing storage location overflow and optimizing space utilization.
Solution: Real-time capacity checking with automatic rejection mechanisms:
Capacity Tracking: Records capacity and current utilization for each location
Trigger Validation: Checks capacity limits before insertion
Volume Calculation: Calculates space requirements based on product volume
Real-time Computation: Dynamically updates location utilization

$$\sum_{p\in Product}(Volume_p \times Quantity_{p,l}) \leq Capacity_l, \quad \forall l \in Location$$

Delay Monitoring

Problem: Timely identification of delayed inbound and outbound operations to improve response speed.
Solution: Automated threshold-based alert system:
Real-time Monitoring: Triggers monitor order status changes
Multi-level Alerts: Sets alerts of varying severity levels
Dynamic Thresholds: Adjusts alert criteria based on historical processing times
Automatic Cleanup: Clears related alerts upon order completion

$$\text{AlertLevel} = \begin{cases} \text{none}, & \text{if } \mathit{\Delta} t \leq T_{\text{normal}} \\ \text{warning}, & \text{if } T_{\text{normal}} < \mathit{\Delta} t \leq T_{\text{critical}} \\ \text{critical}, & \text{if } \mathit{\Delta} t > T_{\text{critical}} \end{cases}$$

Employee Operation Permission Control

Problem: Ensuring employees can only perform operations in assigned warehouses, preventing unauthorized actions.
Solution: Trigger-based permission verification:
Employee-Warehouse Association: Defines employees' working warehouses
Pre-operation Verification: Checks permissions before order creation
Real-time Rejection: Immediately rejects operations with insufficient permissions

Inventory Adjustment Audit Trail

Problem: Complete recording of all inventory changes to meet compliance requirements.
Solution: Comprehensive adjustment recording system:
Dedicated Adjustment Table: Records all inventory changes
Change Classification: Distinguishes types such as count, transfer, write-off, split
Complete History: Records before/after quantities, reasons, and operators
Association Tracking: All changes can be traced to specific personnel

$\forall \Delta Inventory \neq 0, \exists r \in R : \text{record}(t, \Delta I, r, o)$

2.4 Gap Analysis

Existing solutions have limitations:

1. Commercial Systems: High cost, complex implementation
2. Spreadsheets: Poor scalability, data inconsistency
3. Open Source: Incomplete features, limited support

This system addresses these gaps through a balanced approach combining robust data management with practical warehouse operations.

3 METHOD / SYSTEM

3.1 System Architecture Overview

The warehouse management system follows a three-tier architecture with clear separation of concerns:

Presentation Tier (Python CLI) <−−> Application Tier (Business Logic: Procedures, Functions, Views, Triggers) <−−> Data Tier (MySQL Database)

Design Rationale:

Maintainability: Clear separation allows independent development and testing of each layer

Reusability: Business logic in stored procedures can be accessed by multiple front-end applications

Security: Database access can be controlled through stored procedure execution privileges

Scalability: Each tier can be scaled independently based on load requirements

3.2 Database Design Formalization

Core Entities:

– **Warehouse:**

$$W = \big\{ w \mid w = \langle \text{id}, \text{name}, \text{type}, \text{capacity} \rangle \big\}$$

– **Product:**
$$P = \{p \mid p = \langle \text{sku}, \text{name}, \text{volume}, \text{value} \rangle\}$$
– **Inventory:**
$$I = \{i \mid i = \langle w, p, \text{location}, quantity_{\text{total}}, quantity_{\text{available}}, quantity_{\text{locked}} \rangle\}$$
– **Order:**
$$O = \{o \mid o = \langle \text{type}, \text{status}, \text{quantity}, \text{timestamp} \rangle\}$$

Relationships:

Storage Relationship:

–
$$R_{\text{storage}} \subseteq W \times P \times I \quad \text{(Many-to-many via inventory)}$$

Explanation: Links warehouses, products, and inventory records. Each warehouse stores multiple products; each product in multiple warehouses.
Implementation: inventory table with $warehouse_{id}$ and $product_{id}$ foreign keys.

Supply Relationship:

–
$$R_{\text{supply}} \subseteq \text{Supplier} \times P \quad \text{(Many-to-many via supplier\_products)}$$

Explanation: Connects suppliers with products. Multiple suppliers provide same product; suppliers offer multiple products.
Implementation: $supplier_{products}$ table with composite key $(supplier_{id}, product_{id})$.

Employee Relationship:

–
$$R_{\text{employee}} \subseteq \text{Employee} \times W \quad \text{(Many-to-one assignment)}$$

Explanation: Associates employees with assigned warehouses. Each employee to one warehouse; warehouses have multiple employees.
Implementation: $employees.warehouse_{id}$ foreign key to warehouses.

Key Constraints (Mathematical Formulation)

Capacity Constraint:

– $\forall w \in W, \forall l \in \text{Locations}(w) : \quad \sum_{i \in I(w,l)} \text{volume}(p_i) \times quantity_{\text{total}}(i) \leq \text{capacity}(w, l)$

Explanation: Total product volume in location $\leq$ location capacity.
Implementation: $prevent_storage_overflow$ trigger before inventory insert / update.

Inventory Conservation:

– $\forall p \in P : \quad \sum_{w \in W} quantity_{\text{total}}(w, p, t) = \sum_{w \in W} quantity_{\text{total}}(w, p, t_0) + \sum_{o \in O_{\text{in}}(t_0,t)} \text{quantity}(o) - \sum_{o \in O_{\text{out}}(t_0,t)} \text{quantity}(o)$

Explanation: Product inventory changes only through documented orders.
Implementation: ACID transactions ensure all changes via $inbound_{orders}$ / $outbound_{orders}$.

Availability Partitioning:

– $\forall i \in I : \quad quantity_{\text{total}}(i) = quantity_{\text{available}}(i) + quantity_{\text{locked}}(i)$ where $quantity_{\text{available}}(i) \geq 0,\ quantity_{\text{locked}}(i) \geq 0$

Explanation: Total inventory = available + locked quantities.
Implementation: Check $quantity_{total} = quantity_{locked} + quantity_{available}$ in inventory table.

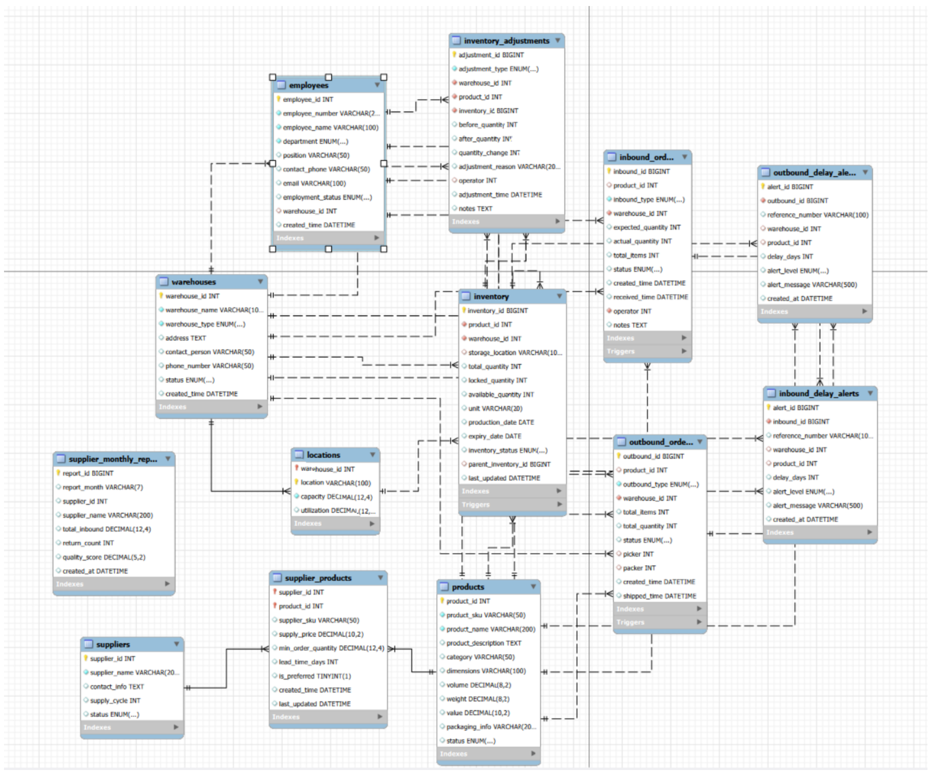


Fig. 1. Database ER Model

Database Schema Implementation

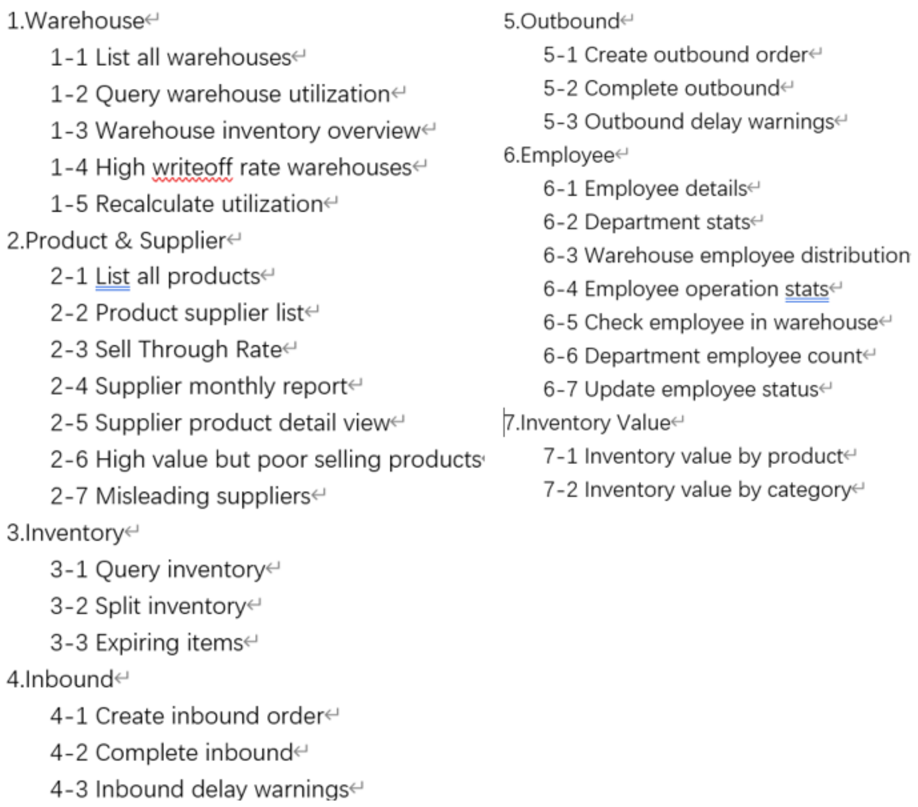
Table 1. Core Tables and Their Purposes

Table	Primary Key	Key Attributes	Purpose
warehouses	*warehouse_id*	*warehouse_type*, status	Warehouse master data
locations	*warehouse_id*, location	capacity, utilization	Storage location management
products	*product_id*	sku, volume, value	Product catalog
inventory	*inventory_id*	*total_quantity*, *available_quantity*, *locked_quantity*	Real-time inventory tracking
inbound_orders	*inbound_id*	*inbound_type*, status, *expected_quantity*	Receiving operations
outbound_orders	*outbound_id*	*outbound_type*, status, *total_quantity*	Shipping operations

3.3 System Architecture Design and Component Interaction:

This warehouse management system adopts a modular architecture design, dividing complex functionalities into seven logically distinct modules, each corresponding to a core business domain within warehouse management. The Warehouse Management module is responsible for physical space and fundamental operational status management, providing infrastructure support for the entire system. The Product and Supplier module focuses on maintaining product catalogs and supply chain relationships, offering the data foundation for procurement and supplier evaluation. As the core of the system, the Inventory Management module handles all specific inventory operations and status tracking, ensuring the accuracy and real-time nature of inventory data. The Inbound and Outbound Management modules standardize the receiving and shipping processes, respectively, forming a complete logistics operation chain. The Employee Management module manages system users and operational permissions, providing security and audit support for all business operations. Finally, the Inventory Value module offers inventory analysis from a financial perspective, bridging operational data with financial management.

The various modules form an organic whole through carefully designed interaction relationships. The Inventory module serves as the data hub, maintaining close data exchange with all other modules: it obtains product information from the Product module, provides storage status to the Warehouse module, offers quantity validation for the Inbound/Outbound modules, records operational history for the Employee module, and contributes valuation data to the Inventory Value module. This hub-and-spoke architecture ensures data consistency and operational coordination. Meanwhile, the inter-module dependencies simulate actual business processes: suppliers provide goods that enter the inbound process, transform into inventory to support outbound operations, all performed by authorized employees and generating performance data, ultimately aggregated into financial value analysis. This design enables the system to naturally support the complete supply chain process from procurement to sales.



1.Warehouse
- 1-1 List all warehouses
- 1-2 Query warehouse utilization
- 1-3 Warehouse inventory overview
- 1-4 High writeoff rate warehouses
- 1-5 Recalculate utilization

2.Product & Supplier
- 2-1 List all products
- 2-2 Product supplier list
- 2-3 Sell Through Rate
- 2-4 Supplier monthly report
- 2-5 Supplier product detail view
- 2-6 High value but poor selling products
- 2-7 Misleading suppliers

3.Inventory
- 3-1 Query inventory
- 3-2 Split inventory
- 3-3 Expiring items

4.Inbound
- 4-1 Create inbound order
- 4-2 Complete inbound
- 4-3 Inbound delay warnings

5.Outbound
- 5-1 Create outbound order
- 5-2 Complete outbound
- 5-3 Outbound delay warnings

6.Employee
- 6-1 Employee details
- 6-2 Department stats
- 6-3 Warehouse employee distribution
- 6-4 Employee operation stats
- 6-5 Check employee in warehouse
- 6-6 Department employee count
- 6-7 Update employee status

7.Inventory Value
- 7-1 Inventory value by product
- 7-2 Inventory value by category

Fig. 2. Operations available in Python CLI

The system's design follows the principles of separation of concerns and clear interfaces, with each module encapsulating specific domain business logic and providing services through standardized database views and stored procedures. This architecture not only reduces system complexity and improves maintainability but also provides a clear path for future functional expansion. Data consistency is ensured through transaction processing and trigger mechanisms, while the modular structure allows the system to operate efficiently as a whole while enabling relatively independent evolution and optimization of individual components. Overall, this architecture meets current warehouse management needs while laying a solid foundation for long-term system development and integration with other enterprise systems.

3.4 Data Flow Patterns:

Read-Heavy Pattern (Analytics): Python Interface → View Query → Optimized Join Execution → Formatted Results
Example: employee-operation-stats-view

Write-Intensive Pattern (Operations): Python Interface → Stored Procedure → Transaction → Multiple Table Updates
Example: split-inventory

Real-time Pattern (Monitoring): Database Event → Trigger Execution → Alert Generation → Alert Table Update
Example: two delay-warning triggers

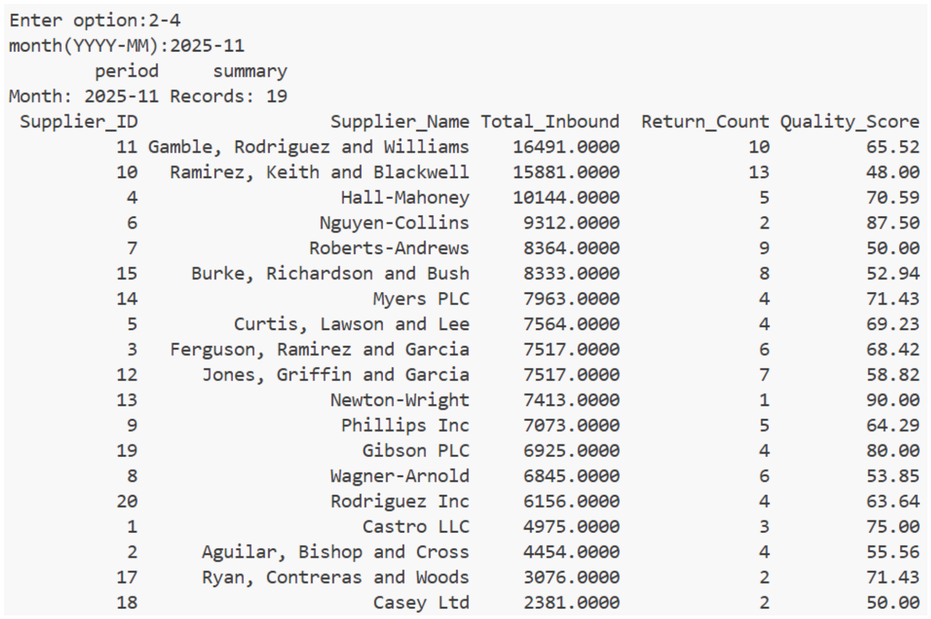
3.5 Algorithmic Approaches

Supplier Performance Analysis Algorithm in supplier_monthly_report

Algorithm Overview The supplier_monthly_report stored procedure implements a multi-dimensional evaluation algorithm for supplier performance analysis. This algorithm aggregates and analyzes inbound data for a specified month to calculate three core performance metrics: total inbound quantity, return count, and quality score.

$$\text{QualityScore} = \begin{cases} 100\%, & \text{if TotalOrders} = 0 \\ \left(1 - \frac{\text{ReturnCount}}{\text{CompletedOrders}}\right) \times 100\%, & \text{otherwise} \end{cases}$$

Sell-through Rate The calculate_sell_through_rate function implements the calculation algorithm for Sell-through Rate, a key business metric. Sell-through



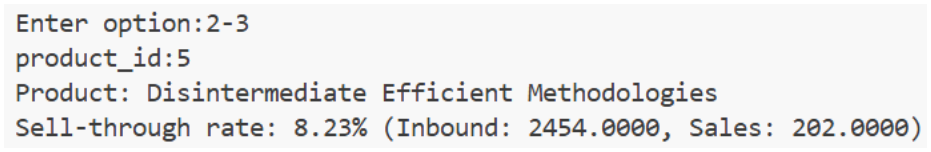
```
Enter option:2-4
month(YYYY-MM):2025-11
        period     summary
Month: 2025-11 Records: 19
 Supplier_ID                 Supplier_Name Total_Inbound  Return_Count Quality_Score
          11 Gamble, Rodriguez and Williams    16491.0000            10         65.52
          10   Ramirez, Keith and Blackwell    15881.0000            13         48.00
           4                  Hall-Mahoney    10144.0000             5         70.59
           6                Nguyen-Collins     9312.0000             2         87.50
           7               Roberts-Andrews     8364.0000             9         50.00
          15     Burke, Richardson and Bush     8333.0000             8         52.94
          14                     Myers PLC     7963.0000             4         71.43
           5          Curtis, Lawson and Lee     7564.0000             4         69.23
           3   Ferguson, Ramirez and Garcia     7517.0000             6         68.42
          12      Jones, Griffin and Garcia     7517.0000             7         58.82
          13                 Newton-Wright     7413.0000             1         90.00
           9                  Phillips Inc     7073.0000             5         64.29
          19                    Gibson PLC     6925.0000             4         80.00
           8                 Wagner-Arnold     6845.0000             6         53.85
          20                 Rodriguez Inc     6156.0000             4         63.64
           1                    Castro LLC     4975.0000             3         75.00
           2      Aguilar, Bishop and Cross     4454.0000             4         55.56
          17      Ryan, Contreras and Woods     3076.0000             2         71.43
          18                    Casey Ltd     2381.0000             2         50.00
```

Fig. 3. supplier_monthly_report Result

Rate measures the conversion efficiency of products from inbound to sales and serves as an important performance indicator for inventory management.

$$\mathrm{STR}(p) = \frac{\displaystyle\sum_{o \in S(p)} q(o)}{\displaystyle\sum_{i \in I(p)} q(i)} \times 100\%$$

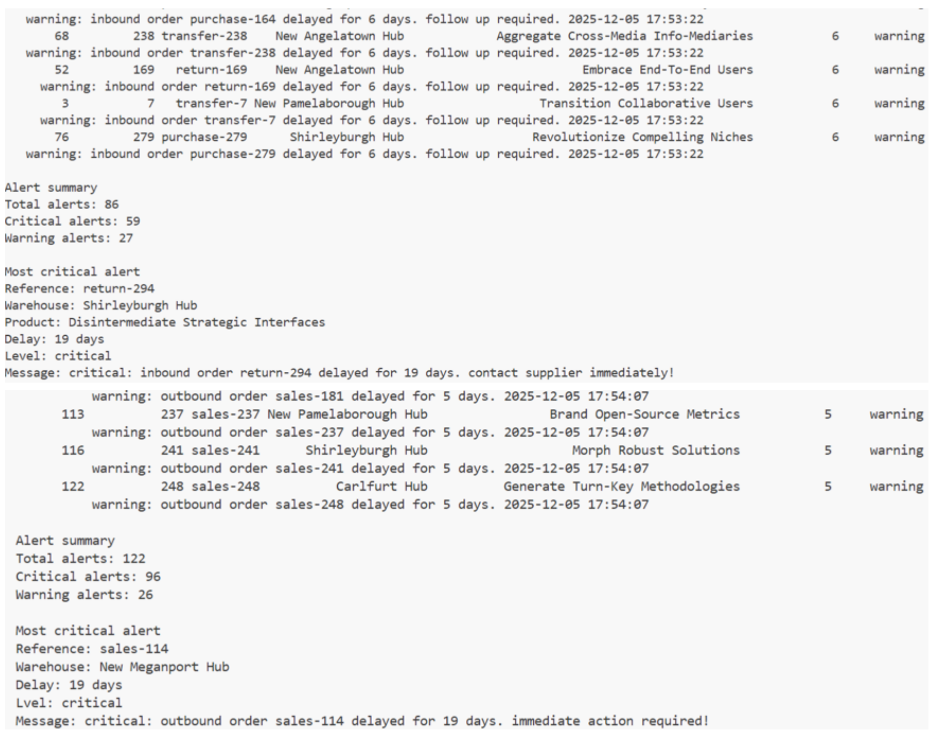
where:
S(p): All shipped sales orders for product
I(p): All completed inbound records for product
q(x): Quantity of order or inbound record



```
Enter option:2-3
product_id:5
Product: Disintermediate Efficient Methodologies
Sell-through rate: 8.23% (Inbound: 2454.0000, Sales: 202.0000)
```

Fig. 4. calculate_sell_through_rate Result

Real-time Delay Detection Algorithm The delay warning trigger system implements a state-based real-time monitoring algorithm. The system continuously monitors status changes of inbound and outbound orders and automatically generates tiered alerts based on delay days.



```
  warning: inbound order purchase-164 delayed for 6 days. follow up required. 2025-12-05 17:53:22
     68        238 transfer-238    New Angelatown Hub            Aggregate Cross-Media Info-Mediaries          6    warning
  warning: inbound order transfer-238 delayed for 6 days. follow up required. 2025-12-05 17:53:22
     52        169   return-169    New Angelatown Hub                        Embrace End-To-End Users          6    warning
    warning: inbound order return-169 delayed for 6 days. follow up required. 2025-12-05 17:53:22
      3          7   transfer-7 New Pamelaborough Hub                   Transition Collaborative Users          6    warning
    warning: inbound order transfer-7 delayed for 6 days. follow up required. 2025-12-05 17:53:22
     76        279 purchase-279      Shirleyburgh Hub                  Revolutionize Compelling Niches          6    warning
  warning: inbound order purchase-279 delayed for 6 days. follow up required. 2025-12-05 17:53:22

Alert summary
Total alerts: 86
Critical alerts: 59
Warning alerts: 27

Most critical alert
Reference: return-294
Warehouse: Shirleyburgh Hub
Product: Disintermediate Strategic Interfaces
Delay: 19 days
Level: critical
Message: critical: inbound order return-294 delayed for 19 days. contact supplier immediately!
```

```
          warning: outbound order sales-181 delayed for 5 days. 2025-12-05 17:54:07
      113         237 sales-237 New Pamelaborough Hub               Brand Open-Source Metrics          5    warning
          warning: outbound order sales-237 delayed for 5 days. 2025-12-05 17:54:07
      116         241 sales-241      Shirleyburgh Hub                  Morph Robust Solutions          5    warning
          warning: outbound order sales-241 delayed for 5 days. 2025-12-05 17:54:07
      122         248 sales-248          Carlfurt Hub          Generate Turn-Key Methodologies          5    warning
          warning: outbound order sales-248 delayed for 5 days. 2025-12-05 17:54:07

 Alert summary
 Total alerts: 122
 Critical alerts: 96
 Warning alerts: 26

 Most critical alert
 Reference: sales-114
 Warehouse: New Meganport Hub
 Delay: 19 days
 Lvel: critical
 Message: critical: outbound order sales-114 delayed for 19 days. immediate action required!
```

Fig. 5. Real-time Delay Detection Algorithm Result

Location Utilization Calculation Location utilization means calculate the volume occupied by products. Each product has a specific volume because they are in the boxes. In this circumstance, we have the possibility to calculate the space that has been used in each location. The quantity of products comes from the inventory table. In each inventory record, there are: product_id, which can be used to join the product table in order to get the volume of the product; warehouse_id and location, which helps find the capacity of the specific location in the warehouse; quantity of the product, offering the number of boxes. Therefore, we can calculate the utilization of a location.

$$\text{Utilization}(w, l) = \sum_{i \in I(w,l)} V(P(i)) \times Q(i)$$

where:
w, l: Location in a given warehouse
$I(w, l)$: The inventory stored in the given location
$V(P(i))$: The volume of the product
$Q(i)$: The quantity of the product in the inventory

High write-off rate warehouses The purpose of this view is to identify warehouses whose product losses are significantly above normal levels. In our system, any product removed from inventory due to damage, expiration, or other non-saleable conditions is recorded as a write-off in the inventory_adjustments table. Each write-off entry includes the warehouse_id and the quantity that has been deducted from the inventory. By aggregating all write-off adjustments for each warehouse, we can compute the total quantity of products that have been written off historically. And we define that warehouses for which the absolute write-off quantity exceeds 0.5% of all received products are classified as high-loss warehouses.

Total write-off quantity in a warehouse:

$$W(w) = \sum_{i \in A(w)} |\Delta Q(i)|$$

where:
$A(w)$: All write-off adjustments of inventory in warehouse w
$Q(i)$: Quantity change in adjustment record i

Total inbound actual quantity:

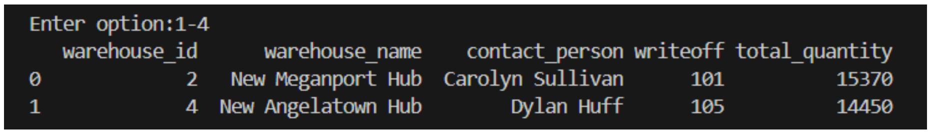
$$T(w) = \sum_{j \in IB(w)} AQ(j)$$

where:
$IB(w)$: All inbound orders of warehouse
$AQ(j)$: Actual received quantity in inbound order

A warehouse is classified as high-loss if:

$$\text{WriteOffRate}(w) = \frac{W(w)}{T(w)} \geq 0.005$$

Misleading Suppliers We notice that, in the supplier-products table, some suppliers providing a specific product are preferred. That means, when the manager of the warehouses considers selecting which supplier for a product, those suppliers preferred are likely to be chosen. However, some of the suppliers are



```
Enter option:1-4
   warehouse_id      warehouse_name    contact_person writeoff total_quantity
0             2   New Meganport Hub  Carolyn Sullivan      101          15370
1             4  New Angelatown Hub        Dylan Huff      105          14450
```

Fig. 6. High Write-Off Rate Warehouses Result

not worth trusted——their products are often returned by the customers. Therefore, it is important to detect these misleading suppliers in the list of preference. We define the suppliers are "misleading", if the total number of inbound entries from a supplier that are flagged as "return" exceeds 20% of the supplier's total inbound transactions.

Number of return-flagged inbound entries from supplier s:

$$R(s) = \sum_{i \in RT(s)} 1$$

Total inbound transactions from supplier s.

$$T(s) = \sum_{j \in IB(s)} 1$$

A supplier is defined as misleading if:

$$\text{Return Rate}(s) = \frac{R(s)}{T(s)} > 0.2$$

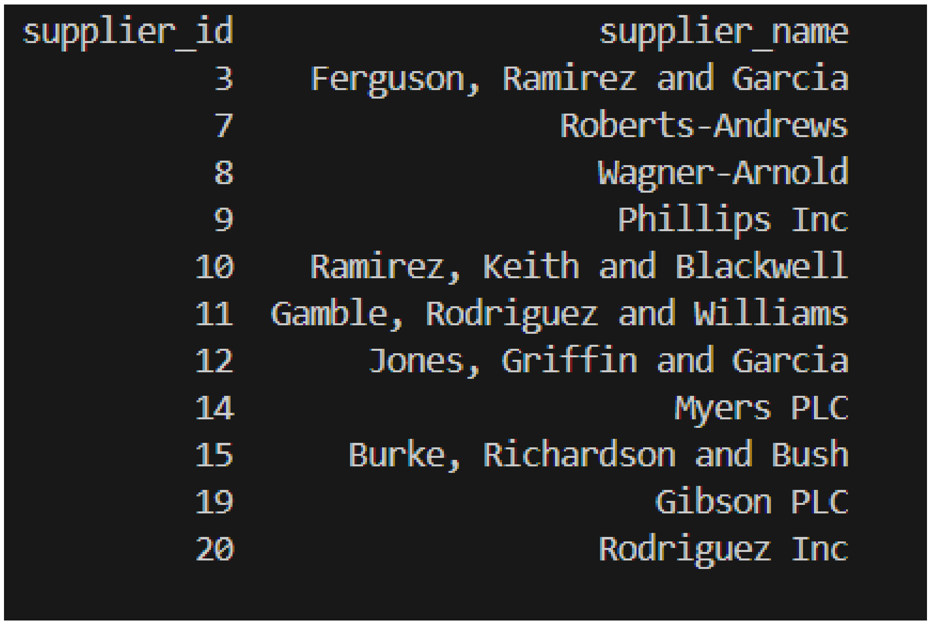
where:
s: A supplier.
$RT(s)$: The set of inbound records from supplier s flagged as "return".
$IB(s)$: The set of all inbound transactions associated with supplier s.

High Value but Poor Selling Products To identify products that may be overstocked or performing poorly, we analyze their total inventory value and recent outbound sales. A product is considered a *high-value poor-selling product* if its total inventory value exceeds the average inventory value of its category, and it has no "shipped" outbound transactions within the past three months.

First, the total inventory value of a product is computed as:

$$V(p) = \sum_{i \in I(p)} \text{val}(p) \times Q(i)$$

where:



```
supplier_id                    supplier_name
          3    Ferguson, Ramirez and Garcia
          7                 Roberts-Andrews
          8                   Wagner-Arnold
          9                    Phillips Inc
         10    Ramirez, Keith and Blackwell
         11  Gamble, Rodriguez and Williams
         12       Jones, Griffin and Garcia
         14                       Myers PLC
         15      Burke, Richardson and Bush
         19                      Gibson PLC
         20                   Rodriguez Inc
```

Fig. 7. Misleading Suppliers Result

p: A product.
$I(p)$: All inventory records associated with product p.
$\text{val}(p)$: The unit value of product p.
$Q(i)$: Quantity in inventory entry i.

The category-level average inventory value is:

$$\overline{V}(c) = \frac{1}{\sum_{P(c)}} \sum_{p \in P(c)} V(p)$$

where:
c: A category.
$P(c)$: Set of all products belonging to category c.
$V(p)$: Total inventory value of product p.

A product qualifies as high-value only if:

$$V(p) > \overline{V}(c)$$

To assess selling performance, we check outbound activity. A product is considered poor-selling if:

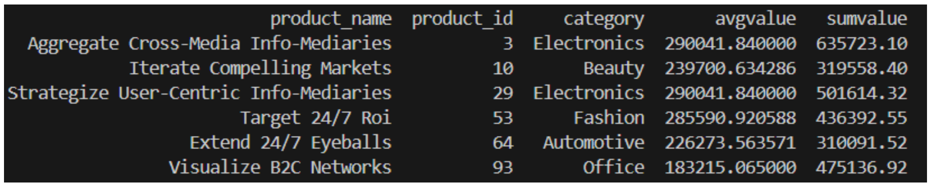
$$\text{Not exists} \quad o \in O(p) \quad \text{such that}$$

$$o.\text{status} = \text{“shipped” and } o.\text{shipped\_time} \geq \text{Today} - 3 \text{ months}$$

where:
$O(p)$: All outbound orders involving product p.

Finally, a product is classified as a high-value poor-selling product if:

$$\left(V(p) > \overline{V}(c)\right) \quad \text{and} \quad \text{No recent shipped outbound orders in 3 months.}$$



```
                      product_name  product_id     category        avgvalue   sumvalue
   Aggregate Cross-Media Info-Mediaries           3  Electronics  290041.840000  635723.10
            Iterate Compelling Markets          10       Beauty  239700.634286  319558.40
Strategize User-Centric Info-Mediaries          29  Electronics  290041.840000  501614.32
                    Target 24/7 Roi          53      Fashion  285590.920588  436392.55
                Extend 24/7 Eyeballs          64   Automotive  226273.563571  310091.52
             Visualize B2C Networks          93       Office  183215.065000  475136.92
```

Fig. 8. High Value but Poor Selling Products Result

4 COMPUTATION RESULTS / EVALUATION

4.1 Comparison with Northwind Database[4]

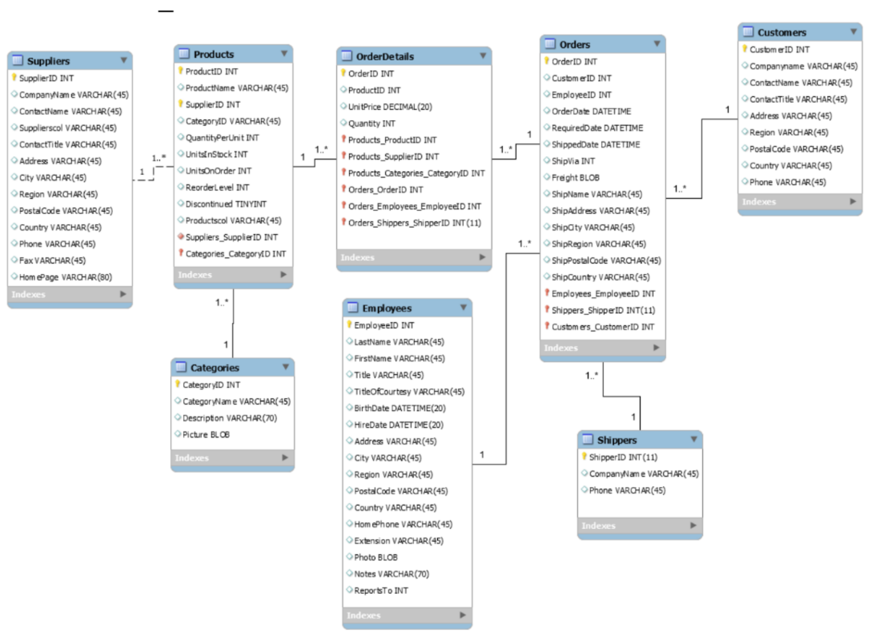


Fig. 9. Northwind Database

For comparison, we select the widely used sample database Northwind Database as a baseline system for inventory and order management. Northwind is a classic teaching-oriented relational database that models products, suppliers, customers, orders and inventory flow.

Advantages of Our Database Design

Compared with classical sample database Northwind, our warehouse management system provides a higher level of operational detail. First, the schema offers more details in warehouse organization, including storage locations and location-based inventory. This enables the system to support real-world warehouse operations such as storing and picking products, which enables workers to find the right location for each inventory.

Second, our system designs the relation inventory adjustments with logs, improving traceability and accountability. Adjustment records not only capture quantity changes of the inventory, but also link operations to the corresponding warehouse, product and employee.

Finally, the system supports a broader range of inventory-related operations, such as inbound, outbound, transfers, and damage reporting. These operations are enforced through triggers, procedures, and integrity constraints to maintain data consistency.

Disadvantages of Our Database Design

Despite its strengths, the system also presents several limitations. Unlike Northwind, which includes customer, order, and shipping modules, our database focuses primarily on warehouse operations and does not reflect customer or sales entities.

Furthermore, the number of attributes per entity is relatively limited, which means that it may lack certain fields which can be typically found in industrial warehouse systems.

Both Strength and Weakness

An important characteristic of our schema is that relations are highly interdependent. On one hand, the complex network of foreign keys and constraints ensures strong data integrity, preventing mismatched or incomplete records. This reflects real warehouse management rules and improves reliability.

On the other hand, this tight coupling increases schema complexity. Highly connected relations may lead to more complicated querying, higher risk of cascading inconsistencies if not carefully managed, and greater difficulty in scaling or modifying the schema in the future. Thus, the design enhances correctness but requires more careful maintenance.

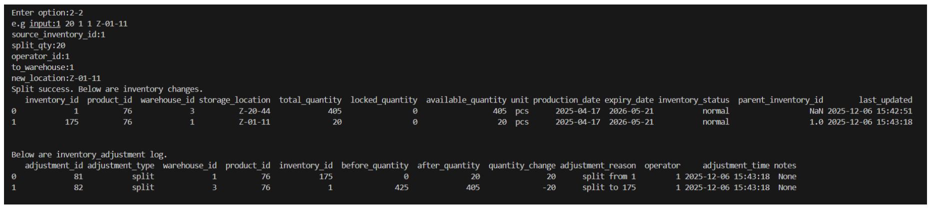
4.2 Capabilities of our Database

In our warehouse management system, inbound and outbound operations ensure accurate recording of goods entering and leaving the warehouse and proper tracking of inventory. Among these operations, these core procedures are essential for maintaining data consistency and complete business operations: split_inventory, create_inbound(outbound), and complete_inbound(outbound).

Split inventory to another warehouse The split_inventory procedure is used when an existing inventory needs to be divided into smaller ones. For example, part of a inventory needs being separated to another warehouse.

Key Operations:

1. Retrieve the original inventory and validate that the split quantity does not exceed available stock.
2. Reduce the quantity of the original inventory.
3. Create a new inventory with a unique inventory_id and the specified split quantity.
4. Record the parent–child batch relationship for auditing and traceability.
5. Update the inventory_adjustment table to record the change.



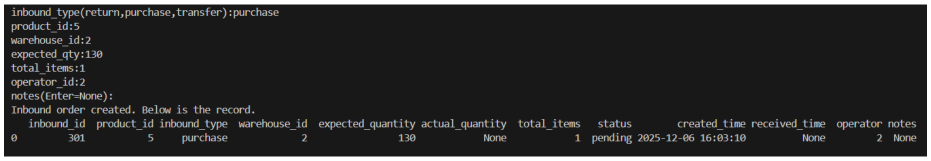
```
Enter option:2-2
e.g input:1 20 1 1 Z-01-11
source_inventory_id:1
split_qty:20
operator_id:1
to_warehouse:1
new_location:Z-01-11
Split success. Below are inventory changes.
   inventory_id  product_id  warehouse_id storage_location  total_quantity  locked_quantity  available_quantity unit production_date expiry_date inventory_status  parent_inventory_id        last_updated
0             1          76             3          Z-20-44             405                0                 405  pcs      2025-04-17  2026-05-21           normal                  NaN 2025-12-06 15:42:51
1           175          76             1          Z-01-11              20                0                  20  pcs      2025-04-17  2026-05-21           normal                  1.0 2025-12-06 15:43:18


Below are inventory_adjustment log.
   adjustment_id adjustment_type  warehouse_id  product_id  inventory_id  before_quantity  after_quantity  quantity_change adjustment_reason  operator     adjustment_time notes
0             81           split             1          76           175                0              20               20      split from 1         1 2025-12-06 15:43:18  None
1             82           split             3          76             1              425             405              -20      split to 175         1 2025-12-06 15:43:18  None
```

Fig. 10. Split Inventory

Create inbound/outbound orders The create_inbound_orders and create_outbound_orders procedures record a new inbound/outbound order prior to the arrival or departure of goods. To avoid repetition, the report only presents the inbound operations.

Key Operations

1. Validate the belong-to relationship between operator and warehouse.
2. Generate a new inbound/outbound record with information provided.
3. Initialize the order status as "pending".



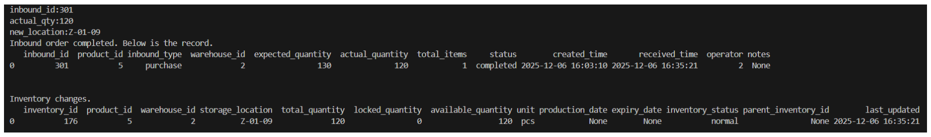
```
inbound_type(return,purchase,transfer):purchase
product_id:5
warehouse_id:2
expected_qty:130
total_items:1
operator_id:2
notes(Enter=None):
Inbound order created. Below is the record.
   inbound_id  product_id inbound_type  warehouse_id  expected_quantity actual_quantity  total_items   status        created_time received_time  operator notes
0         301           5     purchase             2                130            None            1  pending 2025-12-06 16:03:10          None         2  None
```

Fig. 11. Create Inbound

complete_inbound The complete_inbound function finalizes an inbound order once goods are physically received by the warehouse.

Key Operations

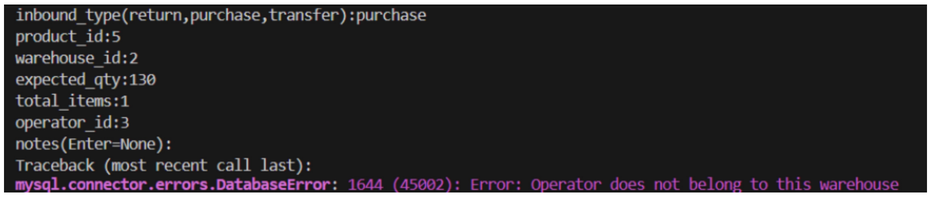
1. Verify the location's capacity is enough to store the new inventory.
2. Create or update inventory based on the received quantity.
3. Update the inbound record's status to "completed".
4. Record timestamps and responsible employee information for auditing.



```
inbound_id:301
actual_qty:120
new_location:Z-01-09
Inbound order completed. Below is the record.
   inbound_id  product_id inbound_type  warehouse_id  expected_quantity  actual_quantity  total_items     status        created_time       received_time  operator notes
0         301           5     purchase             2                130              120            1  completed 2025-12-06 16:03:10 2025-12-06 16:35:21         2  None


Inventory changes.
   inventory_id  product_id  warehouse_id storage_location  total_quantity  locked_quantity  available_quantity unit production_date expiry_date inventory_status parent_inventory_id         last_updated
0           176           5             2          Z-01-09             120                0                 120  pcs            None        None           normal                None 2025-12-06 16:35:21
```

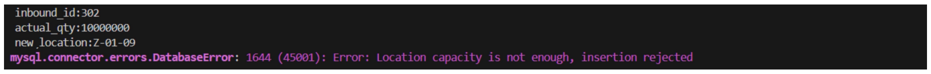
Fig. 12. Complete Inbound

The two triggers below ensure proper data insertion. The first trigger happens when creating inbound order. If the operator does not belong to the given warehouse, it will be triggered to prevent the create inbound. And the second trigger happens when completing inbound. Before the actual quantity of products are stored in the given location, the trigger would calculate whether the capacity of the location is enough to store the inbound products. If so, the inbound order will be rejected.



```
inbound_type(return,purchase,transfer):purchase
product_id:5
warehouse_id:2
expected_qty:130
total_items:1
operator_id:3
notes(Enter=None):
Traceback (most recent call last):
mysql.connector.errors.DatabaseError: 1644 (45002): Error: Operator does not belong to this warehouse
```

Fig. 13. Trigger:Operator does not belong to the corresponding warehouse



```
inbound_id:302
actual_qty:10000000
new_location:Z-01-09
mysql.connector.errors.DatabaseError: 1644 (45001): Error: Location capacity is not enough, insertion rejected
```

Fig. 14. Trigger:Location's capacity is not enough to store the inbound quantity

5 CONCLUSION

This project successfully designed and implemented a MySQL-based warehouse management system that integrates advanced database concepts with real-world operational needs. The system addresses key challenges in modern logistics, including real-time inventory tracking, multi-warehouse coordination, expiration management, capacity control, and employee permission enforcement, through a robust and scalable architecture.

By employing a modular, three-tier design with clear separation of concerns, the system ensures maintainability, reusability, and security. The database schema, supported by stored procedures, triggers, and views, enforces data integrity and supports complex analytical queries. Key functionalities such as supplier performance analysis, sell-through rate calculation, and real-time delay detection demonstrate the system's capability to provide actionable insights for warehouse managers.

Compared to the classic Northwind database, our system offers greater operational granularity, enhanced traceability, and support for a wider range of warehouse processes. However, it also presents limitations, such as a narrower business scope and increased schema complexity due to tight coupling.

Overall, this project validates that a thoughtfully designed open-source database system can deliver enterprise-grade warehouse management capabilities without the high costs and vendor lock-in associated with commercial solutions.

5.1 Future Work

Potential enhancements include integrating machine learning for demand forecasting, adding a web-based graphical user interface, extending support for customer and sales modules, and implementing real-time dashboards for operational monitoring. Further scalability testing and performance optimization in cloud environments would also be valuable for industrial adoption.

6 APPENDIX

6.1 Contributions

Parts that each member did 50% : Proposal, Software Requirements Specification, Database Design Document, Presentation, Final Paper, Peer Assessment, Python Interface

Qixuan Yuan) Operations: high_writeoff_rate_warehouses, list_products, get_product_suppliers,high_value_products_with_poor_selling, misleading_suppliers, query_inventory, split_inventory, expiring_items, create_inbound_order, complete_inbound, create_outbound_order, complete_outbound, inventory_value_by_product

Shaoxiang Qu: Operations: calculate_sell_through_rate, generate_supplier_monthly_report, view_supplier_products_detail, view_inbound_delay_warnings, view_outbound_delay_warnings, view_employee_details, view_department_employee_stats, view_warehouse_employee_distribution, view_employee_operation_stats, check_employee_in_warehouse, get_department_employee_count, update_employee_status, category_inventory_overview, trigger_delay_alerts;
README file

6.2 Weekly Project Timeline

Table 2. Weekly Project Timeline Table.

Week	Key Accomplishments	Obstacles Encountered	Strategies Employed
Week 1 (20/10 - 26/10)	Make decision of project topic	Figuring out project topic	Brainstrom, browsing for information
Week 2 (27/10 - 2/11)	Finished Software Requirements Specification (SRS) document	Writing SRS; Design schema for database; MySQL Deploying	Learn SRS pattern through document: Requirements-Engineering-Chapter-4.1-4.5.pdf; Brainstrom, browsing for information
Week 3 (3/11 - 9/11)	Finished Project Database Design Document	Design schema for database, making changes during the process	Brainstrom, browsing for information; Following instructions on MySQL official website
Week 4 (10/11 - 16/11)	Finish writing part of views, functions, procedures, and triggers	Designing views, functions, procedures, and triggers; Adjust Schema accordingly	Brainstrom, browsing for information; Following instructions on MySQL official website
Week 5 (17/11 - 23/11)	Finish writing part of views, functions, procedures, and triggers; Finish writing part of interface	Designing views, functions, procedures, and triggers; Adjust Schema accordingly; Writing simple interface	Brainstrom, browsing for information; Following instructions on MySQL official website
Week 6 (24/11 - 30/11)	Finish writing all the views, functions, procedures, and triggers; Finish writing interface Finalizing & Testing the project	Finalizing & Testing views, functions, procedures, and triggers; Finalizing & Testing interface; Writing final report	Brainstrom, browsing for information about how to write final report; Following instructions on MySQL official website
Week 7 (1/12 - 7 / 12)	Finish writing final report	Writing final report	Browsing for information and learn about how to write final report

References

1. Brinch, M.: Understanding the value of big data in supply chain management and its business processes. International Journal of Operations & Production Management **38**(7), 1589–1614 (2018). https://doi.org/10.1108/IJOPM-05-2017-0268
2. Perera, S., Pinto, A., Sewmini, H., Ulugalathenne, A., Thelijjagoda, S., Karunarathna, N.: Influence of IoT on Warehouse Management Performance in the Global Context: A Critical Literature Review. In: 2nd International

Conference on Sustainable & Digital Business (ICSDB), pp. 118–132 (2023). https://doi.org/10.54389/mlep9597

3. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse operation: A comprehensive review. European Journal of Operational Research **177**(1), 1–21 (2006). https://doi.org/10.1016/j.ejor.2006.02.025
4. Microsoft: Downloading sample databases **2**017 https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/downloading-sample-databases