**Wholesale Management System**

**A Project Report**

Submitted by

Adwitaa Singh(24MCA20005)

Angel Taneja(24MCA20011)

In partial fulfilment for the award of the degree of

MASTERS OF COMPUTER APPLICATION



Chandigarh University

OCTOBER 2024

**INDEX**

|  |  |
| --- | --- |
| **Topic** | **Page No.** |
| Introduction | 3 |
| Objectives | 3 |
| PL/SQL Code | 6-11 |
| Reports | 11 |
| Testing | 12 |
| Result | 12 |
| Conclusion | 16 |

**Introduction**

A **Wholesale Management System (WMS)** in PL/SQL could serve as an efficient way to manage wholesale distribution activities, including inventory control, order management, supplier and customer tracking, and financial reporting.

The operational procedures of wholesale firms can be automated and streamlined with the help of a Wholesale Management System (WMS). The system is a vital tool for handling financial transactions, tracking supplier and customer connections, processing orders, and managing inventories. Given the size and complexity of wholesale operations, maintaining efficiency, minimizing errors, and facilitating effective communication amongst various stakeholders all depend on a well-designed management system.

By automating essential tasks and reducing manual intervention, the PL/SQL WMS project aims to improve the operational efficiency of wholesale businesses. This project highlights the value of database-driven solutions in enhancing efficiency, accuracy, and scalability for wholesale organizations in addition to showcasing the usefulness of PL/SQL in a real-world commercial setting.

**1. Problem Statement**

The wholesale industry needs a robust system to manage various processes, such as inventory control, customer and supplier management, and sales. The current system, based on manual processes or outdated software, results in data redundancy, inefficiency, and human errors. The wholesale business is looking for an automated solution that can streamline their operations and handle large volumes of data efficiently.

**2. Objective**

To design and implement a **Wholesale Management System** using PL/SQL that automates and streamlines the wholesale processes, including:

* **Inventory Management:** Managing stock levels, tracking product movements.
* **Supplier and Customer Management:** Recording supplier details, order history, and customer data.
* **Order Processing:** Managing customer orders, supplier orders, and sales transactions.
* **Financial Management:** Handling payments, tracking accounts payable and receivable.
* **Reporting and Analytics:** Generating various reports to assist management in decision-making.

**3. System Requirements**

**Functional Requirements:**

* **Inventory Module:**
  + Add, update, and delete products.
  + Track stock levels, reorder products automatically when stock falls below a threshold.
  + Record product details like supplier, price, and quantity.
* **Supplier and Customer Module:**
  + Add, update, and remove supplier/customer details.
  + Track supplier/customer order history.
  + View payment and order statuses.
* **Order Module:**
  + Create customer orders, and generate invoices.
  + Create purchase orders for suppliers.
  + Handle order fulfillment, shipping, and returns.
* **Financial Module:**
  + Track sales and purchases.
  + Generate bills, handle accounts payable and receivable.
  + Calculate revenue, profit margins, and taxes.
* **Reporting Module:**
  + Generate reports on inventory, sales, supplier performance, and customer orders.
  + Display financial reports like profit and loss, balance sheets, and expense reports.

**Non-Functional Requirements:**

* High performance in handling large volumes of transactions.
* Secure access to the system with role-based permissions.
* Ensure data integrity and backup capabilities.

**4. System Design**

**Entity-Relationship Diagram (ERD)**

The system consists of the following main entities:

* **Product**: Contains product details, stock levels, and related supplier.
* **Supplier**: Contains supplier details and purchase orders.
* **Customer**: Contains customer details and sales orders.
* **Order**: Represents sales and purchase orders, with related payment details.
* **Payment**: Tracks payments for orders.
* **User**: Manages login and roles for admin, manager, and employee users.

**Table Design:**

* **PRODUCTS** table to store product information.
* **SUPPLIERS** table to store supplier details.
* **CUSTOMERS** table to store customer information.
* **ORDERS** table for both purchase and sales orders.
* **PAYMENTS** table to track transactions.
* **USERS** table to manage user roles and access.

**PL/SQL Stored Procedures:**

1. **Inventory Management:**
   * Procedure to add/update/delete products.
   * Function to check stock levels and trigger reordering.
2. **Order Processing:**
   * Procedure to create customer and supplier orders.
   * Procedure to update order statuses (e.g., shipped, delivered).
3. **Financial Management:**
   * Procedure to record payments, track pending dues.
   * Function to calculate total sales for a given period.
4. **Reports Generation:**
   * Procedure to generate sales, inventory, and financial reports.

**5. Implementation:**

The system would use **Oracle PL/SQL** for the back-end logic, with stored procedures handling core business logic. A simple front-end interface, either a web-based application or a desktop client, would interact with the PL/SQL procedures via APIs or direct database access.

**Sample PL/SQL Code for Creating an Order:**

Creating a **Wholesale Management System (WMS)** using PL/SQL can be an exciting project that covers database design, PL/SQL procedures, triggers, and functions to handle the core operations of a wholesale business. Below is a detailed project outline that you can use as a blueprint to build such a system.

**1. Database Design**

**Table Structure:**

CREATE TABLE suppliers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

contact VARCHAR(100),

address TEXT

);

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

description TEXT,

price DECIMAL(10, 2) NOT NULL,

quantity INT NOT NULL,

supplier\_id INT,

FOREIGN KEY (supplier\_id) REFERENCES suppliers(id)

);

CREATE TABLE customers (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE,

phone VARCHAR(20)

);

CREATE TABLE orders (

id INT AUTO\_INCREMENT PRIMARY KEY,

product\_id INT,

customer\_id INT,

quantity INT NOT NULL,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('Pending', 'Shipped', 'Delivered', 'Cancelled') DEFAULT 'Pending',

FOREIGN KEY (product\_id) REFERENCES products(id),

FOREIGN KEY (customer\_id) REFERENCES customers(id)

);

**Insert Data:**

INSERT INTO suppliers (name, contact, address) VALUES

('ABC Supplies', '123-456-7890', '123 Main St, Citytown'),

('XYZ Wholesalers', '987-654-3210', '456 Side Ave, Metropolis');

INSERT INTO products (name, description, price, quantity, supplier\_id) VALUES

('Product A', 'High-quality product A', 10.50, 100, 1),

('Product B', 'Durable product B', 15.75, 50, 2);

INSERT INTO customers (name, email, phone) VALUES

('John Doe', 'john.doe@example.com', '555-1234'),

('Jane Smith', 'jane.smith@example.com', '555-5678');

INSERT INTO orders (product\_id, customer\_id, quantity, status) VALUES

(1, 1, 10, 'Pending'),

(2, 2, 5, 'Shipped');

**Basic SQL Queries:**

SELECT \* FROM products;

SELECT orders.id, customers.name AS customer\_name, products.name AS product\_name,

orders.quantity, orders.status, orders.order\_date

FROM orders

JOIN customers ON orders.customer\_id = customers.id

JOIN products ON orders.product\_id = products.id;

**Update Queries:**

UPDATE products

SET quantity = quantity - 5

WHERE id = 1;

UPDATE orders

SET status = 'Delivered'

WHERE id = 1;

**Delete Date:**

DELETE FROM products

WHERE id = 2;

DELETE FROM customers

WHERE id = 2;

**Filtered Queries:**

SELECT \* FROM products

WHERE quantity < 50;

SELECT \* FROM orders

WHERE status = 'Pending';

**Aggregation:**

SELECT SUM(quantity) AS total\_stock FROM products;

SELECT COUNT(\*) AS total\_customers FROM customers;

**Reports:**

SELECT

P.id AS 'Product ID',

P.name AS 'Product Name',

P.description AS 'Description',

P.price AS 'Price',

P.quantity AS 'Current Stock Level',

P.supplier\_id AS 'Supplier ID',

S.name AS 'Supplier Name',

S.contact AS 'Supplier Contact',

S.address AS 'Supplier Address',

O.id AS 'Order ID',

O.quantity AS 'Order Quantity',

O.order\_date AS 'Order Date',

O.status AS 'Order Status',

C.name AS 'Customer Name',

C.email AS 'Customer Email',

C.phone AS 'Customer Phone'

FROM

products P

LEFT JOIN

suppliers S ON P.supplier\_id = S.id

LEFT JOIN

orders O ON P.id = O.product\_id

LEFT JOIN

customers C ON O.customer\_id = C.id

ORDER BY

P.name, O.order\_date;

**4. Testing the System**

* **Unit Testing**: Test individual stored procedures, functions, and triggers.
* **Integration Testing**: Ensure the overall workflow (placing an order, updating inventory, generating reports) functions as expected.
* **Performance Testing**: Test with large datasets to ensure the system can handle high volumes of transactions efficiently.

**5. Future Enhancements**

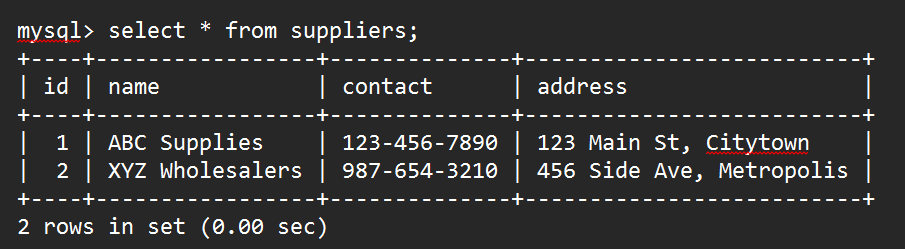
* **Integration with third-party systems** like payment gateways, logistics, or accounting software.
* **Analytics and Forecasting**: Using data analytics to predict demand and optimize inventory management.
* **Mobile Application Integration**: Develop a mobile app for on-the-go order management and reporting.

**6. Results and Benefits:**

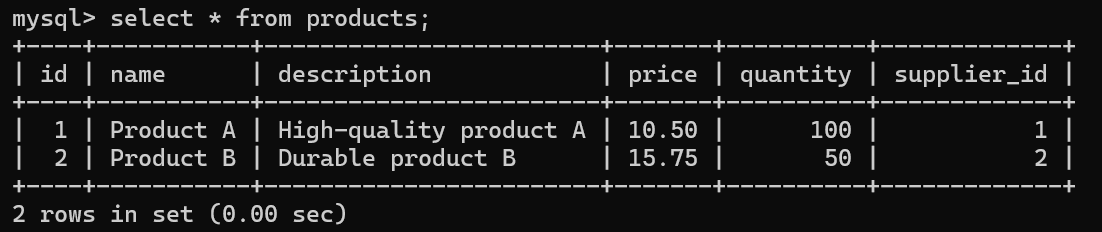
* The **Wholesale Management System** significantly reduces manual intervention, leading to fewer errors in data entry.
* It offers **real-time inventory tracking**, allowing the business to reorder products before they run out.
* The **financial module** keeps accurate records of sales and payments, streamlining accounting tasks.
* **Automated reporting** provides valuable insights for better decision-making.

**OUTPUT:**

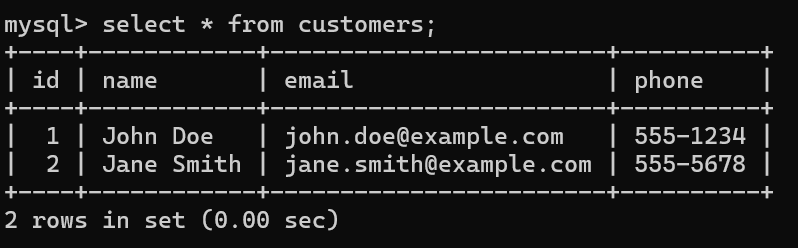
**Suppliers:**



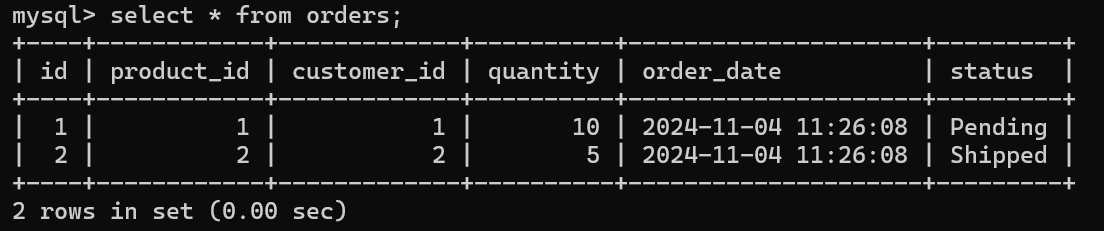
**Products:**

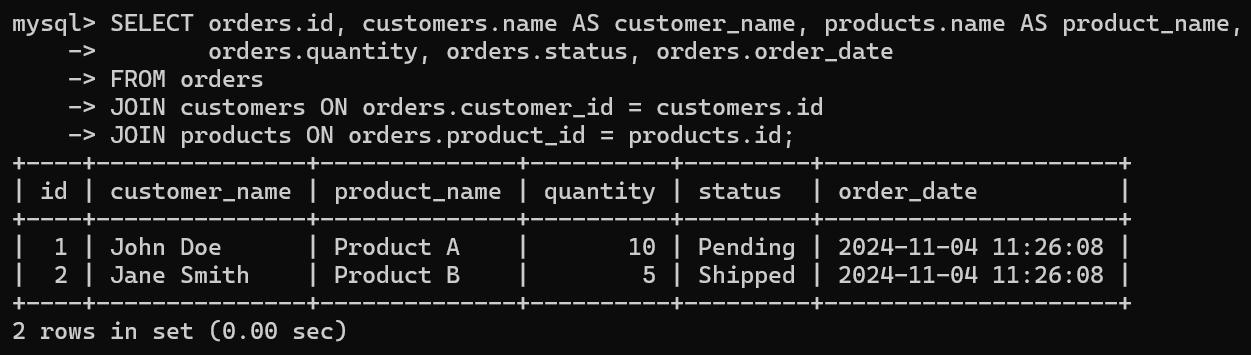
****

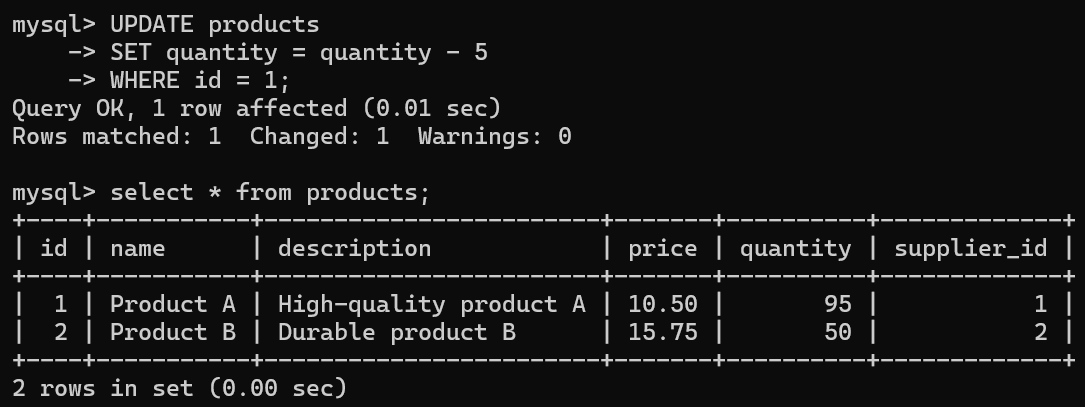
**Customers:**

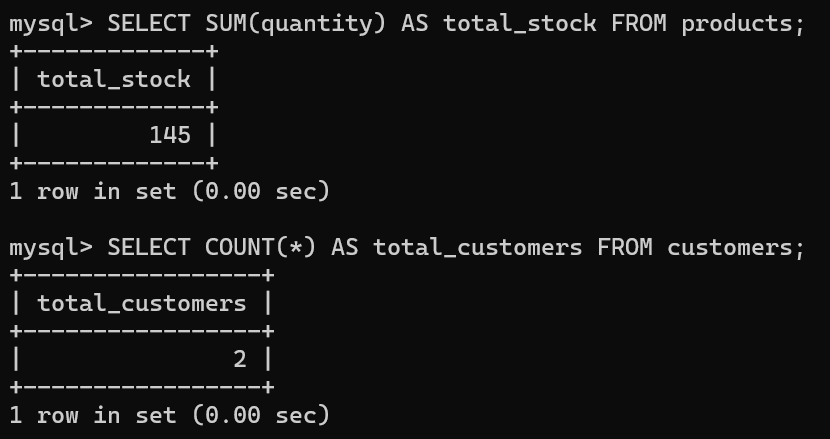
****

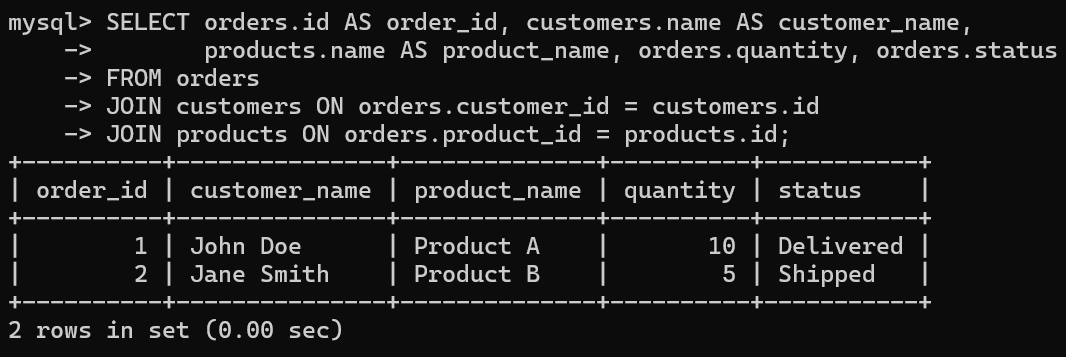
**Orders:**

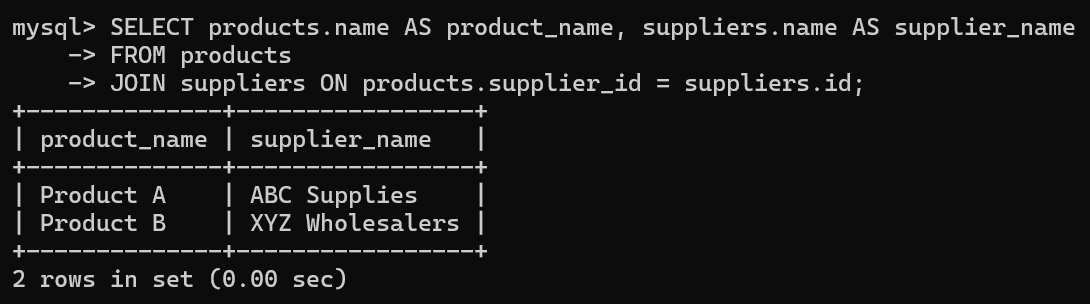
****

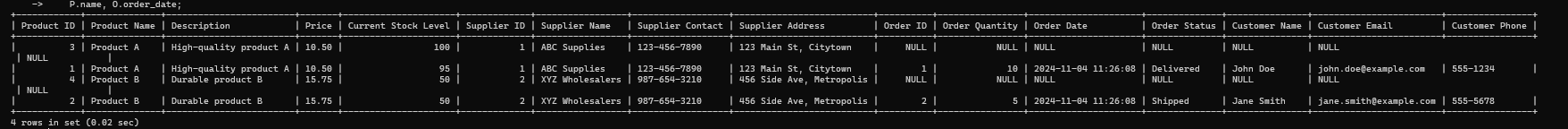
****





****





**CONCLUSION:**

By automating essential tasks and reducing manual intervention, the PL/SQL WMS project aims to improve the operational efficiency of wholesale businesses. This project highlights the value of database-driven solutions in enhancing efficiency, accuracy, and scalability for wholesale organizations in addition to showcasing the usefulness of PL/SQL in a real-world commercial setting.