DBMS PROJECT OVERVIEW

1)Title : **Godown Management system for daily consumer goods**

2)Team Members:
i)Vineet V Pai PES1UG22CS698
ii)Vishwanath S B PES1UG22CS704

3)Purpose :
The purpose of the project is to showcase the transactions that take place in a godown(warehouse) especially for daily consumer goods. The priority focus was aimed at showcasing the way stocks of a product changes in a godown with new supplies and new purchases happening affecting the total availability.

4)**Scope of the Project**

1. **Primary Functionality**
    - **Inventory Management**:
        - Maintain a real-time record of stock availability for each product.
        - Handle stock updates based on incoming supplies and outgoing sales or dispatches.
    - **Transaction Management**:
        - Record supply details, including supplier information, quantity supplied, and dates.
        - Record purchase details, including customer information, quantities sold, and dates.
2. **Product Information**
    - Track detailed information about each product, including:
        - Product name, category, and availability.
        - Expiration dates and manufacturing details.
        - Current stock levels and reorder thresholds.
3. **Reports**

    Generate reports on:

- Any purchasings done
- Any sales carried out
- Highest selling price per unit of a product.

5) Description :

Managing inventory and transactions in a godown (warehouse) for daily consumer goods is a complex and time-sensitive task. Traditional methods, such as manual record-keeping or disconnected systems, are prone to errors, inefficiencies, and delays. These challenges lead to problems such as inaccurate stock tracking, overstocking or understocking, wastage due to expired goods, and difficulties in analyzing sales and supply trends.

The absence of an integrated system creates the following specific challenges:
1. Inefficient Inventory Management: Difficulty in maintaining real-time stock levels and tracking stock changes due to supplies and purchases.
2. Limited Visibility of Product Details: Lack of accurate and updated information on product expiration dates, reorder thresholds, and manufacturing details.
3. Manual Transaction Tracking: Recording supply and sales transactions manually is time-consuming and increases the risk of errors.
4. Delayed Decision-Making: Without automated alerts or analytics, identifying low-stock items or expired products is delayed, leading to stockouts or wastage.
5. Lack of Insights: Difficulty in generating reports for trend analysis, supplier performance, or product demand.

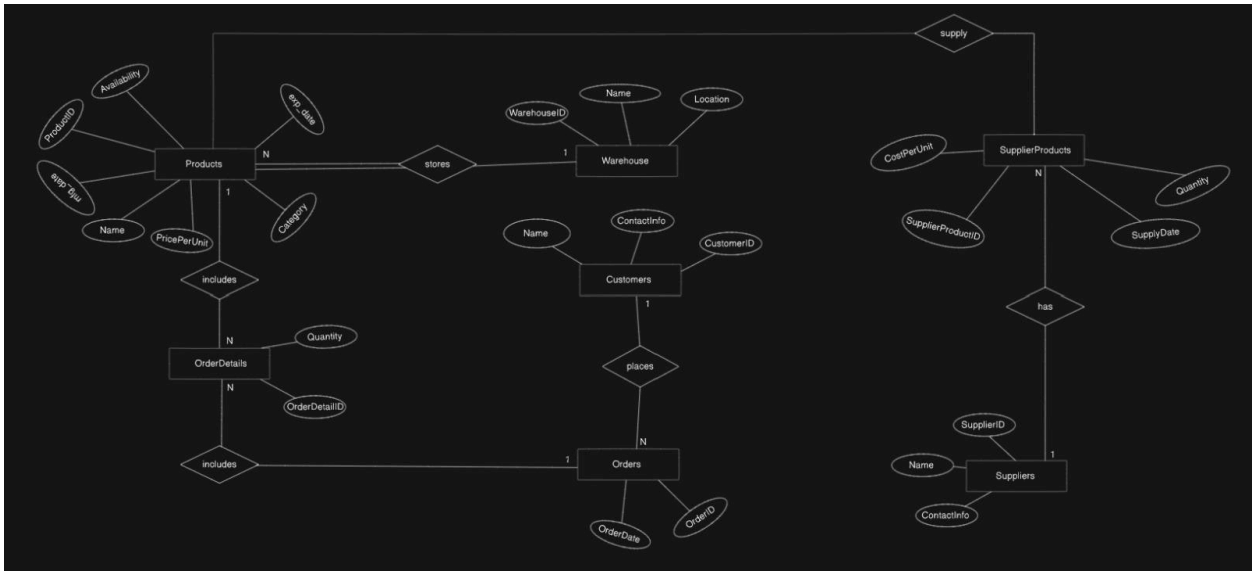
These issues impact the efficiency and profitability of warehouse operations, highlighting the need for an automated solution.

This project aims to address these challenges by developing an integrated Godown Management System that streamlines inventory management, automates transaction tracking, and provides actionable insights through analytics and reporting. The solution will enhance operational efficiency, reduce wastage, and support informed decision-making for daily consumer goods warehouses.

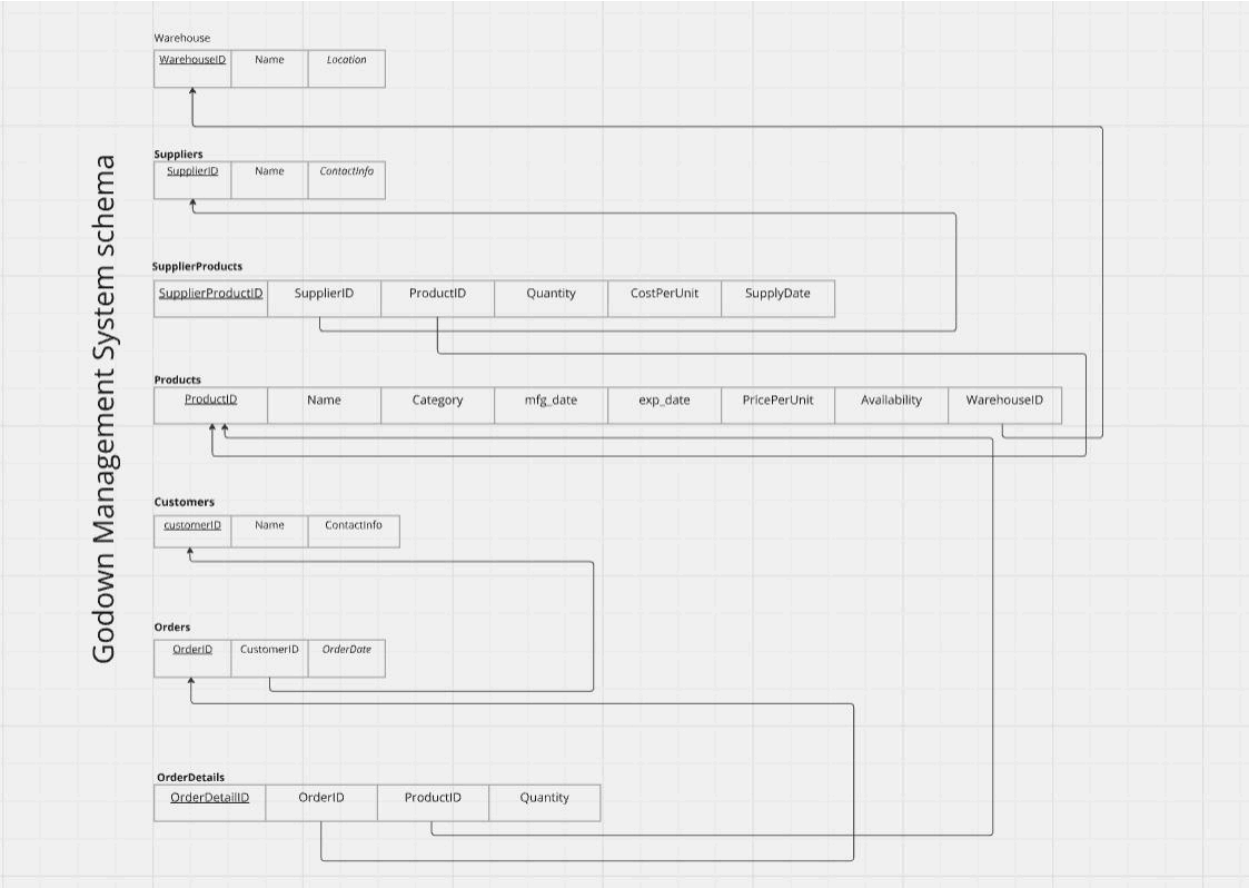
5) Use-Case View

1. **Stock Entry and Updates**
   Warehouse stock tracking personnel can enter new stock or update existing inventory records.
2. **Goods Receipt and Dispatch**
   Tracks incoming and outgoing goods, ensuring accurate inventory records.
3. **Inventory Tracking and Monitoring**
   Provides real-time stock levels and generates low-stock alerts.

6)ER DIAGRAM



7)RELATIONAL SCHEMA



8)QUERIES

i) Creating Database :

```
mysql> create database Godown_management;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| classl             |
| collegedays        |
| company            |
| company_new        |
| fest_database      |
| godown_management  |
| information_schema |
| mysql              |
| performance_schema |
| schooldb           |
| simpledb           |
| studentdb          |
| sys                |
+--------------------+
13 rows in set (0.00 sec)
```

ii) Creating tables

```
mysql> use godown_management;
Database changed
mysql> CREATE TABLE Warehouses (
    ->     WarehouseID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(50) NOT NULL,
    ->     Location VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Products (
    ->     ProductID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100) NOT NULL,
    ->     Category VARCHAR(50) NOT NULL,
    ->     mfg_date DATE NOT NULL,
    ->     exp_date DATE NOT NULL CHECK (exp_date > mfg_date),  -- Ensure expiration date is after manufacturing date
    ->     Availability INT DEFAULT 0,
    ->     PricePerUnit DECIMAL(10, 2) NOT NULL,
    ->     WarehouseID INT,
    ->     FOREIGN KEY (WarehouseID) REFERENCES Warehouses(WarehouseID) ON DELETE SET NULL
    -> );
ERROR 3813 (HY000): Column check constraint 'products_chk_1' references other column.
mysql> CREATE TABLE Products (
    ->     ProductID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100) NOT NULL,
    ->     Category VARCHAR(50) NOT NULL,
    ->     mfg_date DATE NOT NULL,
    ->     exp_date DATE NOT NULL,
    ->     PricePerUnit DECIMAL(10, 2) NOT NULL,     -- Selling price per unit
    ->     Availability INT DEFAULT 0,
    ->     WarehouseID INT,
    ->     FOREIGN KEY (WarehouseID) REFERENCES Warehouses(WarehouseID) ON DELETE SET NULL
    -> );
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> CREATE TABLE Suppliers (
    ->     SupplierID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100) NOT NULL,
    ->     ContactInfo VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE SupplierProducts (
    ->     SupplierProductID INT PRIMARY KEY AUTO_INCREMENT,
    ->     SupplierID INT,
    ->     ProductID INT,
    ->     Quantity INT NOT NULL,
    ->     CostPerUnit DECIMAL(10, 2) NOT NULL,      -- Cost per unit from the supplier
    ->     SupplyDate DATE NOT NULL,
    ->     FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID) ON DELETE CASCADE,
    ->     FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE TABLE Customers (
    ->     CustomerID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100) NOT NULL,
    ->     ContactInfo VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Orders (
    ->     OrderID INT PRIMARY KEY AUTO_INCREMENT,
    ->     CustomerID INT,
    ->     OrderDate DATE NOT NULL,
    ->     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> CREATE TABLE OrderDetails (
    ->     OrderDetailID INT PRIMARY KEY AUTO_INCREMENT,
    ->     OrderID INT,
    ->     ProductID INT,
    ->     Quantity INT NOT NULL,
    ->     FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) ON DELETE CASCADE,
    ->     FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> DELIMITER //
mysql> CREATE TRIGGER after_supply_insert
    -> AFTER INSERT ON SupplierProducts
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Products
    ->     SET Availability = Availability + NEW.Quantity
    ->     WHERE ProductID = NEW.ProductID;
    -> END;
    -> //
Query OK, 0 rows affected (0.04 sec)

mysql> DELIMITER ;
mysql> DELIMITER //
mysql> CREATE TRIGGER after_order_insert
    -> AFTER INSERT ON OrderDetails
    -> FOR EACH ROW
    -> BEGIN
    ->     UPDATE Products
    ->     SET Availability = Availability - NEW.Quantity
    ->     WHERE ProductID = NEW.ProductID;
    -> END;
    -> //
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
```

iii) Integrated queries to list products:

```python
@app.route("/products")
def list_products():
    query = """
        SELECT p.ProductID, p.Name, p.Category, p.PricePerUnit, p.Availability,
               p.mfg_date, p.exp_date, w.Name AS WarehouseName
        FROM Products p
        LEFT JOIN Warehouses w ON p.WarehouseID = w.WarehouseID;
    """
    products = run_query(query)
    return render_template("products.html", products=products)
```

Products

NAME	CATEGORY	PRICE PER UNIT	AVAILABILITY	MANUFACTURE DATE	EXPIRY DATE	WAREHOUSE
Urad Dal	Non perishable Foods	78.00	240	2024-11-01	2024-12-07	A

iv)Query to list out supplier details:

```
@app.route("/sell")
def supplier_product_map():
    # Query for Supplier-Product Relationship Map
    supplier_query = """
        SELECT s.Name AS SupplierName, p.Availability, p.Name AS ProductName, p.Category
        FROM Suppliers s
        JOIN SupplierProducts sp ON s.SupplierID = sp.SupplierID
        JOIN Products p ON sp.ProductID = p.ProductID
        ORDER BY s.Name, p.Name;
    """
    supplier_product_data = run_query(supplier_query)
```

Supplier-Product Relationship Map

Supplier Name	Availability	Product Name	Category
Agro foods	240	Urad Dal	Non perishable Foods

v)Query to list out order details:

```
# Query for Order Details
order_details_query = """
    SELECT c.Name AS CustomerName, o.OrderDate, od.Quantity, p.Name AS ProductName
    FROM OrderDetails od
    JOIN Orders o ON od.OrderID = o.OrderID
    JOIN Customers c ON o.CustomerID = c.CustomerID
    JOIN Products p ON od.ProductID = p.ProductID
    ORDER BY o.OrderDate DESC, c.Name;
"""
order_details_data = run_query(order_details_query)
```

Order Details

Customer Name	Order Date	Quantity	Product Name
Bob	2024-11-15	5	Urad Dal

vi)query to insert into order details and update stock quantity

```
# Check if products are available and insert order details
for product_id, quantity in zip(products, quantities):
    # Check product availability
    check_availability_query = "SELECT Availability FROM Products WHERE ProductID = %s"
    cursor.execute(check_availability_query, (product_id,))
    available_quantity = cursor.fetchone()

    if available_quantity and int(quantity) <= available_quantity[0]:
        # Insert order details into OrderDetails table
        insert_order_details_query = "INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES (%s, %s, %s)"
        cursor.execute(insert_order_details_query, (order_id, product_id, int(quantity)))

        # Update product availability only once
        update_availability_query = "UPDATE Products SET Availability = Availability - %s WHERE ProductID = (%s)/2"
        cursor.execute(update_availability_query, (int(quantity), product_id))
```

vii)query to list out customer details

```
# Get all products for the order form
products_query = "SELECT ProductID, Name FROM Products WHERE Availability > 0"
products = run_query(products_query)

# Get all customers for the customer dropdown
customers_query = "SELECT CustomerID, Name FROM Customers"
customers = run_query(customers_query)

return render_template("orders.html", products=products, customers=customers)
```

viii)query to get product details from warehouse

```
# Query to get products by warehouse
query = """
    SELECT w.name AS WarehouseName, w.location AS WarehouseLocation,
           p.ProductID, p.Name AS ProductName, p.Category, p.mfg_date,
           p.exp_date, p.PricePerUnit, p.Availability
    FROM Warehouses w
    LEFT JOIN Products p ON w.WarehouseID = p.WarehouseID
    ORDER BY w.name;
"""
warehouse_products = run_query(query)
```

viii)Queries to insert new supplier and new product and existing product stock

```
# Insert Product
product_query = """
    INSERT INTO Products (Name, Category, mfg_date, exp_date, PricePerUnit, Availability, WarehouseID)
    VALUES (%s, %s, %s, %s, %s, %s, %s)
"""
cursor.execute(product_query, (product_name, category, mfg_date, exp_date, price_per_unit, availability, warehouse_id))
# Get ProductID of newly added product
product_id = cursor.lastrowid

# Insert into SupplierProducts table
supplier_product_query = """
    INSERT INTO SupplierProducts (SupplierID, ProductID, Quantity, CostPerUnit, SupplyDate)
    VALUES (%s, %s, %s, %s, %s)
"""
cursor.execute(supplier_product_query, (supplier_id, product_id, quantity, cost_per_unit, supply_date))

# Commit transaction
connection.commit()
```

Add Supplier and Products

Supplier Information

Supplier Name:

Agro foods

Contact Info:

8088273401

Product Information

Product Name:

Urad Dal

Category:

Non perishable Foods

Quantity:

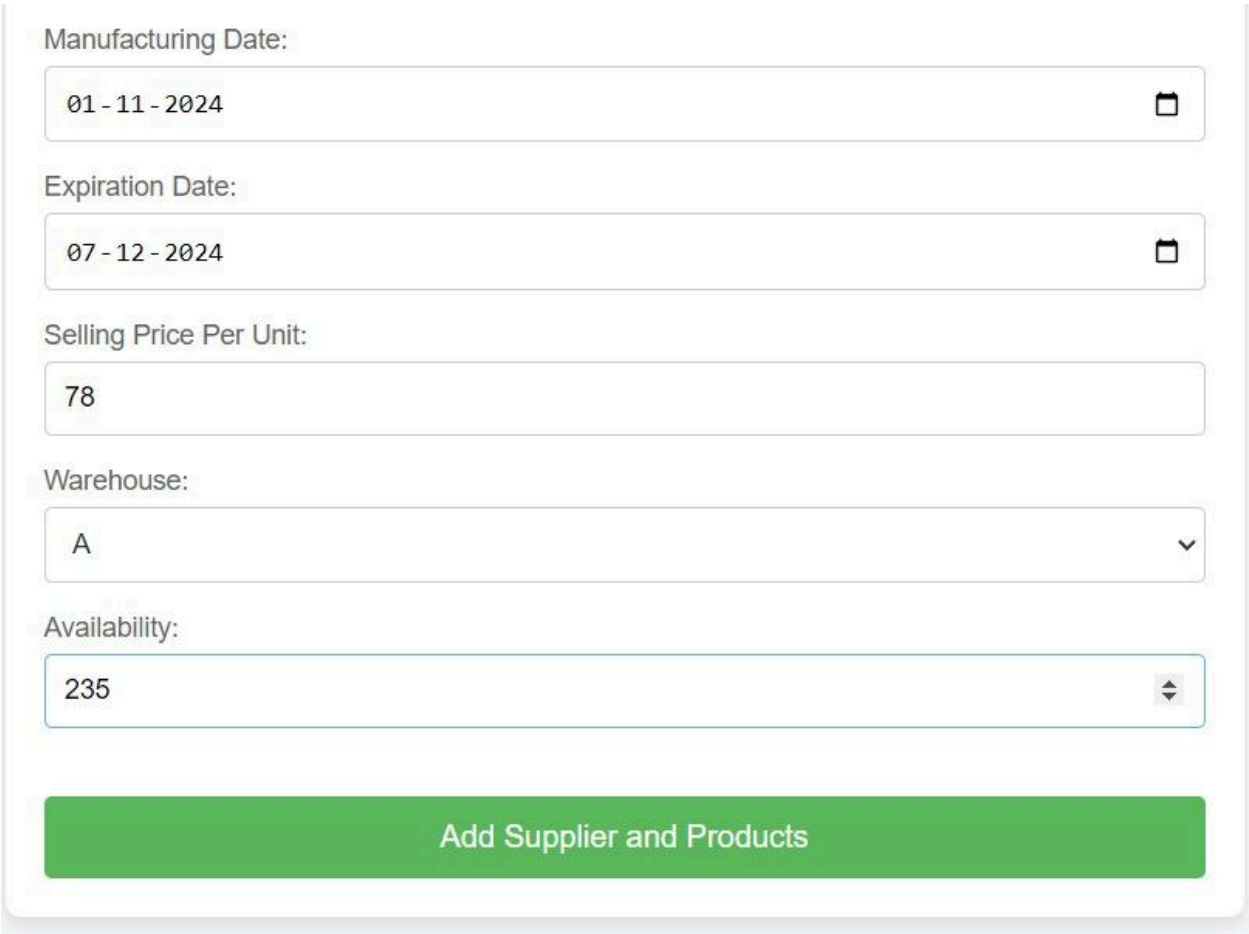
235

Cost Per Unit:

67

Supply Date:

14 - 11 - 2024



ix)Procedure to get products

```
mysql> CREATE PROCEDURE GetProducts()
    -> BEGIN
    ->     SELECT p.ProductID, p.Name, p.Category, p.PricePerUnit, p.Availability,
    ->            p.mfg_date, p.exp_date, w.Name AS WarehouseName
    ->     FROM Products p
    ->     LEFT JOIN Warehouses w ON p.WarehouseID = w.WarehouseID;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
mysql> call procedure getproducts();
```

x)Trigger to update stocks

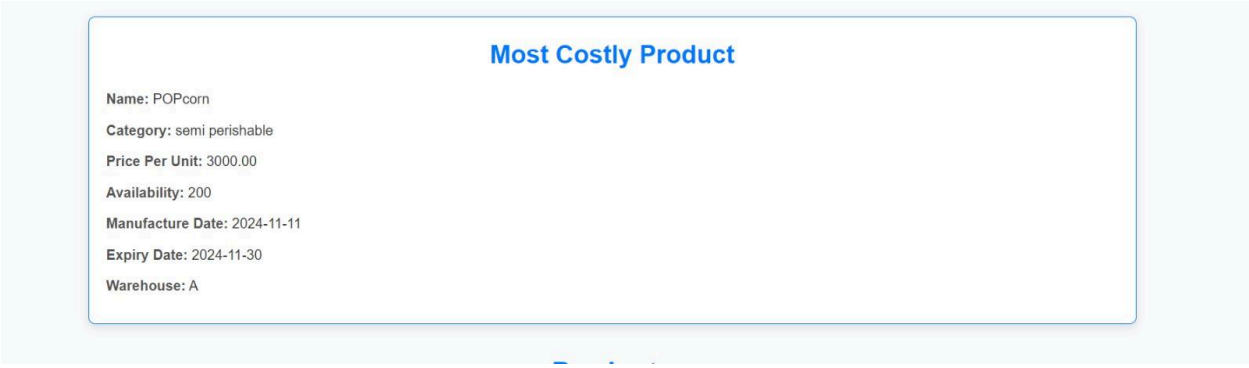
```
mysql> USE godown_management;
Database changed
mysql> CALL GETPRODUCTS;
+-----------+----------+----------------------+--------------+--------------+------------+------------+---------------+
| ProductID | Name     | Category             | PricePerUnit | Availability | mfg_date   | exp_date   | WarehouseName |
+-----------+----------+----------------------+--------------+--------------+------------+------------+---------------+
|         9 | Urad Dal | Non perishable Foods |        78.00 |           70 | 2024-11-01 | 2024-12-07 | A             |
|        10 | dal      | perishable           |        75.00 |           90 | 2024-11-01 | 2024-12-07 | A             |
|        13 | mugdal   | perishable           |        10.00 |          100 | 2024-11-02 | 2024-12-07 | B             |
+-----------+----------+----------------------+--------------+--------------+------------+------------+---------------+
3 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)
```

xi) aggregate function to display the most expensive product available for sales

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE GetMostCostlyProduct()
    -> BEGIN
    ->     SELECT
    ->         p.*,
    ->         w.Name AS WarehouseName
    ->     FROM
    ->         Products p
    ->     LEFT JOIN
    ->         Warehouses w ON p.WarehouseID = w.WarehouseID
    ->     WHERE
    ->         p.PricePerUnit = (SELECT MAX(PricePerUnit) FROM Products);
    -> END //
Query OK, 0 rows affected (0.02 sec)

mysql> DELIMITER ;
```



xii)Warehouse reports