Grocery Store DBMS – Project Report

Introduction

The Grocery Store Database Management System (DBMS) is designed to manage products, customers, and orders in a grocery store. It ensures data consistency, easy CRUD operations, and useful business reports such as sales, top-selling products, and inventory tracking.

Objectives

- Store customer and admin details securely.
- Manage product inventory (stock, pricing, categories).
- Handle customer orders and order details.
- Provide CRUD operations for users, products, and orders.
- Generate business reports like sales, revenue, and low-stock alerts.

ER Diagram

Entities: Users, Products, Orders, Order_Items
Relationships:
- A user can place multiple orders (1 → Many).
- An order can have multiple items (1 → Many).
- Each order item is linked to one product (Many → 1).

Database Schema

```
-- Users Table
CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    role ENUM('customer', 'admin') DEFAULT 'customer',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Products Table
CREATE TABLE products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10,2) NOT NULL,
    stock INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Orders Table
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total_amount DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

-- Order Items Table
CREATE TABLE order_items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(product_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

CRUD Operations

```
-- Insert New Product
INSERT INTO products (name, category, price, stock)
VALUES ('Butter 500g', 'Dairy', 120.00, 25);

-- Update Stock
UPDATE products SET stock = stock - 2 WHERE product_id = 1;

-- Delete Order
DELETE FROM orders WHERE order_id = 3;

-- View Users
SELECT user_id, name, email, role FROM users;
```

Reports

```
-- Low Stock Products
SELECT name, stock FROM products WHERE stock < 10;

-- Orders by Customer
SELECT u.name, o.order_id, o.total_amount, o.order_date
FROM orders o
JOIN users u ON o.user_id = u.user_id
WHERE u.name = 'Priya Sharma';

-- Daily Sales
SELECT DATE(order_date) AS order_day, SUM(total_amount) AS daily_sales
FROM orders
GROUP BY DATE(order_date)
ORDER BY order_day DESC;

-- Top-Selling Products
SELECT p.name, SUM(oi.quantity) AS total_sold
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY p.product_id
ORDER BY total_sold DESC
LIMIT 3;
```

Conclusion

This Grocery Store DBMS provides an efficient way to manage customer data, product inventory, and sales records. The system supports CRUD operations and generates business insights through SQL reports. It can be extended into a full application using Java/Python/PHP for real-world use.