CS-304 Database System 4(3+1)

Final Project Report

Submitted To :

Mr. Khurram Shahzad

Submitted by:

Name	REG no.
Moosa Raza	22-cs-044

Section : B

Department of Computer Science
HITEC University, TAXILA

Contents

1. Inventory Management System: 3
1.1. Description 3
1.2. System Overview 3
1.3. Database Design 3
1.4. Backend Logic (Python - Flask) 4
1.5. Frontend Design (HTML, CSS, Bootstrap) 4
1.6. Workflow of the System 5
1.7. Key Features and Benefits 5
1.8. Explanation of 3NF Normalization 6
1.9. Steps for Applying 3NF on the Tables 6
1.9.1. Categories Table 6
1.9.2. Suppliers Table 6
1.9.3. Customers Table 6
1.9.4. Locations Table 6
1.9.5. Products Table 7
1.9.6. Transactions Table 7
1.9.7. Users Table 7
1.10. How 3NF is Maintained 7
1.11. Enhanced Entity Relationship Diagram: 8
1.12. Conclusion: 9

1. Inventory Management System:

1.1. Description

The **Inventory Management System (IMS)** is a comprehensive solution designed to streamline the management of products, transactions, suppliers, customers, and other related data for businesses. The system is built with an integrated approach, leveraging **SQL for database management**, **Python (Flask)** for backend logic, and **HTML/CSS with Bootstrap** for the frontend.

1.2. System Overview

The IMS provides functionality to manage key aspects of inventory and product transactions, such as:

- **Product Management**: View, add, update, and delete products.
- **Inventory Valuation**: Keep track of stock levels and calculate the total value of inventory.
- **Transaction History**: Record sales, purchases, and transfers of products.
- **Reporting**: Generate various reports, such as inventory valuation, transaction history, and low stock products.
- **User Management**: Admin, Manager, and Employee roles to control access to different parts of the system.

The system is intended for use by businesses that need to track their products, manage suppliers and customers, and maintain detailed transaction histories.

1.3. Database Design

The project starts with a well-organized database schema created using SQL, consisting of several tables that hold key business data:

- **Categories Table**: Stores product categories such as "Electronics", "Furniture", and "Clothing".
- **Suppliers Table**: Contains information about suppliers, including names, contact information, and addresses.
- **Customers Table**: Stores customer details, including their name, contact information, and address.

- **Locations Table**: Manages multiple storage or store locations where products can be stored or sold.
- **Products Table**: This central table stores detailed information about each product, including its name, cost price, selling price, quantity, category, and supplier details.
- **Transactions Table**: Tracks all product transactions, whether it's a sale, purchase, transfer, or adjustment. This table is crucial for understanding inventory movement.
- **Users Table**: Manages user accounts, including admin, manager, and employee roles, each with different permissions and access levels.

Additionally, several **views** are created to make reporting and data access more efficient:

- **LowStockProducts View**: Displays products with stock levels lower than the minimum required quantity.
- **ProductSalesHistory View**: Provides a historical view of product sales, including quantities sold and total sales amounts.
- **InventoryValuation View**: Computes the value of the current inventory by multiplying the quantity of products with their respective cost prices.
- **TransactionHistory View**: Displays a history of all transactions, including details like product names, transaction types (purchase/sale), and customer information.

1.4. Backend Logic (Python - Flask)

The backend of the IMS is developed using **Python** and the **Flask framework**. Flask handles HTTP requests and processes the data, interacting with the SQL database for CRUD operations (Create, Read, Update, Delete). Here's an overview of the backend logic:

- **Database Connection**: A Python function is used to connect to the SQL Server database. The connection is then used to execute SQL queries and retrieve data.
- **Product Management**: Routes are defined to handle operations related to products, such as listing all products, displaying product details, and editing product information.
- **Transaction Handling**: The system allows adding and viewing transactions. It also computes the total value of each transaction based on quantity and unit price.
- **Report Generation**: Flask serves pages that show inventory valuation, transaction history, and low stock products based on queries run against the SQL views.
- **User Management**: The backend ensures that only authorized users (based on roles) can access certain parts of the system, such as editing product details or deleting records.

1.5. Frontend Design (HTML, CSS, Bootstrap)

The **frontend** of the IMS is built using **HTML** for structuring content, **CSS** for styling, and **Bootstrap** for a responsive design. The frontend is responsible for displaying the information fetched from the backend and offering interactive functionality.

Key frontend features include:

- **Product List Page**: Displays all products, with the ability to search, view, edit, or delete them. This page uses a table layout to present the data in a structured manner.
- **Product Details Page**: When a user clicks on a product, they are shown a detailed page with product information such as description, cost price, selling price, and current stock levels.
- **Transaction History Page**: This page shows a comprehensive list of all transactions, detailing the type of transaction (purchase, sale, etc.), product involved, quantities, and customer information.
- **Reports**: Various reports like **Inventory Valuation**, **Transaction History**, and **Low Stock Products** are displayed in table format for easy review and analysis.
- **Navigation**: The frontend includes navigation links to easily move between different sections of the system (products, transactions, reports, etc.).
- **Forms and Modals**: Forms are used for adding and editing products or transactions, while modals are used for confirming actions like deleting records.

The system's interface is designed to be intuitive and user-friendly, with responsive layouts that work on desktop and mobile devices.

1.6. Workflow of the System

1. **Login**: Users first log into the system, where access is granted based on their roles (Admin, Manager, Employee).
2. **Product Management**: The admin or manager can add new products, edit existing ones, and delete products. They can also manage product categories and suppliers.
3. **Transactions**: Sales, purchases, and transfers of products are logged as transactions. The system automatically updates stock levels based on the transaction type.
4. **Inventory Tracking**: The system keeps track of stock levels and alerts users about products that are low in stock.
5. **Reports**: Users can generate reports such as **Inventory Valuation**, **Product Sales History**, and **Low Stock Products** to monitor the performance of the business and make informed decisions.
6. **User Management**: Admins can manage users and their roles to control who has access to what functionalities within the system.

1.7. Key Features and Benefits

- **Efficient Inventory Tracking**: The system helps businesses track products, quantities, and stock values in real-time.
- **Automated Transaction Logging**: All product movements are logged automatically, reducing the risk of errors.
- **Role-Based Access Control**: Users are assigned different roles, ensuring that only authorized personnel can access sensitive data.

- **Easy Reporting**: The system generates various reports that provide insights into inventory levels, sales performance, and customer behavior.
- **User-Friendly Interface**: The system is designed with a clean and responsive interface that makes it easy for users to manage their inventory and transactions.

1.8. Explanation of 3NF Normalization

The **Third Normal Form (3NF)** ensures that:

1. **Elimination of Partial Dependencies**: A non-prime attribute (an attribute not part of a candidate key) must not depend on a part of any candidate key.
2. **Elimination of Transitive Dependencies**: A non-prime attribute must not depend on another non-prime attribute.
3. **Dependency on Primary Key**: Every non-prime attribute must depend directly on the whole primary key.

Below is how each table satisfies 3NF:

1.9. Steps for Applying 3NF on the Tables

1.9.1. Categories Table

- **Attributes**: category_id, name, description
- **Primary Key**: category_id
- All non-prime attributes (name, description) depend only on category_id.
- No partial or transitive dependencies exist, so it satisfies 3NF.

1.9.2. Suppliers Table

- **Attributes**: supplier_id, name, contact_name, email, phone_number, address
- **Primary Key**: supplier_id
- All non-prime attributes depend only on supplier_id.
- No partial or transitive dependencies exist, so it satisfies 3NF.

1.9.3. Customers Table

- **Attributes**: customer_id, name, contact_name, email, phone_number, address
- **Primary Key**: customer_id
- All non-prime attributes depend only on customer_id.
- No partial or transitive dependencies exist, so it satisfies 3NF.

1.9.4. Locations Table

- **Attributes**: location_id, name, address

- **Primary Key**: location_id
- All non-prime attributes depend only on location_id.
- No partial or transitive dependencies exist, so it satisfies 3NF.

1.9.5. Products Table

- **Attributes**: product_id, name, description, category_id, supplier_id, cost_price, selling_price, current_quantity, minimum_quantity, reorder_quantity, unit, image
- **Primary Key**: product_id
- **Foreign Keys**: category_id (to Categories), supplier_id (to Suppliers)
- All non-prime attributes depend only on product_id.
- The foreign keys (category_id and supplier_id) ensure that attributes like name or description of the category or supplier do not appear redundantly in Products, avoiding transitive dependencies.
- Satisfies 3NF.

1.9.6. Transactions Table

- **Attributes**: transaction_id, product_id, transaction_type, quantity, transaction_date, unit_price, location_id, customer_id
- **Primary Key**: transaction_id
- **Foreign Keys**: product_id (to Products), location_id (to Locations), customer_id (to Customers)
- All non-prime attributes (transaction_type, quantity, transaction_date, etc.) depend only on transaction_id.
- Attributes like product_name or customer_name are not stored directly here; instead, they are referenced through foreign keys, ensuring no transitive dependencies.
- Satisfies 3NF.

1.9.7. Users Table

- **Attributes**: user_id, username, password, role
- **Primary Key**: user_id
- All non-prime attributes depend only on user_id.
- The constraint on role (Admin, Manager, Employee) avoids redundancy and ensures no transitive dependencies.
- Satisfies 3NF.

1.10. How 3NF is Maintained

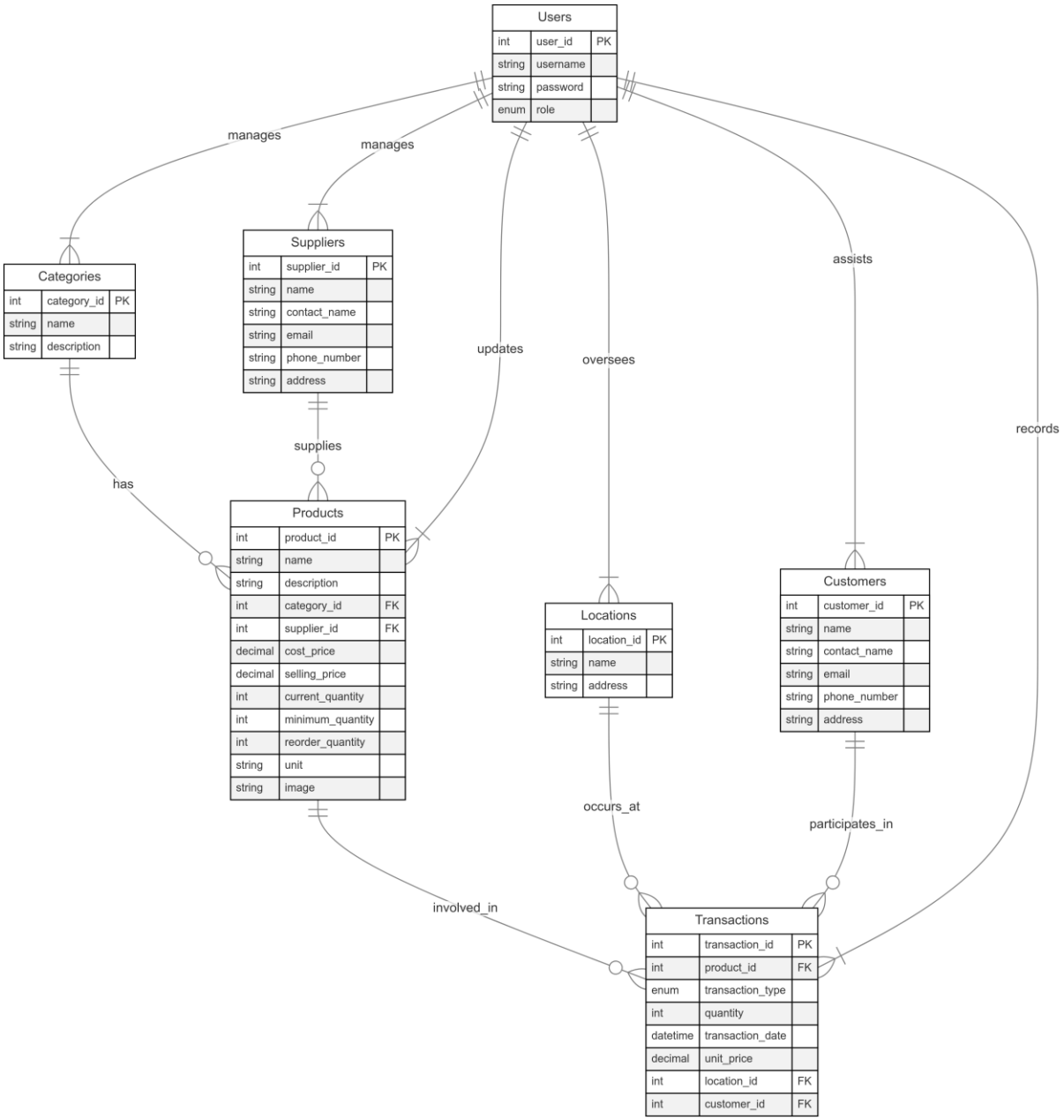
For all tables:

1. **No Partial Dependencies**: Each non-prime attribute depends on the entire primary key (or a candidate key if applicable).

2. **No Transitive Dependencies**: No non-prime attribute depends indirectly on the primary key via another non-prime attribute.
3. **Dependency on Primary Key**: All attributes are directly related to the primary key.

By enforcing foreign keys and splitting data into multiple related tables (e.g., Products referencing Categories and Suppliers), redundancy is minimized, and the database adheres to 3NF. This ensures data consistency, reduces anomalies, and maintains a logical structure.

1.11. Enhanced Entity Relationship Diagram:



1.12. Conclusion:

In conclusion, this Inventory Management System (IMS) project provides an integrated solution for managing products, transactions, suppliers, customers, and other essential data for businesses. The system leverages a well-structured SQL database, with tables for categories, suppliers, customers, products, transactions, and users, all interconnected by foreign key relationships. The backend is developed using Python and Flask, which handles database operations, user authentication, and report generation. The frontend, built with HTML, CSS, and Bootstrap, offers an intuitive, user-friendly interface to interact with the system. With efficient inventory tracking, automated transaction logging, and role-based access control, the system ensures real-time inventory updates, error reduction, and streamlined operations. The project also adheres to the Third Normal Form (3NF) normalization rules, ensuring the database structure is efficient, consistent, and free of redundancy. Through its features and design, the IMS enables businesses to make informed decisions, improve workflow, and maintain data integrity across all operations.