**Medicine Inventory Management System**

|  |  |
| --- | --- |
| **Name** | **Id** |
| Farah Sameh Amin | 231000120 |
| Sandy Ibrahim Twadros | 231000813 |
| Rodaina Mohamed Abdelaziz | 231000739 |
| Mariam Ezzeldin | 231000933 |

Supervisors: Doctor Mohamed Elsayeh

TA. Habiba Mohamed Ibrahim

**Table of Contents**

Contents

[**1.** **Introduction** 3](#_Toc198839034)

[**2.** **ER-Diagram** 3](#_Toc198839035)

[**I.** **Entities and Relationships Description** 4](#_Toc198839036)

[**II.** **Relationships** 4](#_Toc198839037)

[**3.** **Relational Schema** 5](#_Toc198839038)

[**4.** **SQL IMPLEMENTATION** 6](#_Toc198839039)

[**i.** **Create table statements** 6](#_Toc198839040)

[**ii.** **Insert Sample Data** 7](#_Toc198839041)

[**iii.** **Trigger for bill calculation** 9](#_Toc198839042)

[**5.** **Queries and sample output** 9](#_Toc198839043)

[**6.Conclusion** 10](#_Toc198839044)

# **Introduction**

The objective of this project is to design a relational database for a “Medicine Inventory Management System”. It includes all the operational functions supporting a healthcare environment like managing patient prescriptions, maintaining inventory, and billing patients.

The reason why this system has been chosen is because managing medicines in hospitals, clinics, and pharmacies is “critical”. Prescribing errors, medicine stockouts, and billing discrepancies pose potential danger from an operational health care system. A reliable database system, accurate and efficient structure will improve patient safety.

Information about doctors and patients, diseases, medicines, prescriptions and billing will be stored in the databases. They will also allow answering complex queries of extracting information, computation of the most prescribed medicine, and filtering by specialty. Triggering mechanisms for computation will promote automatic billing reducing manual effort on ensuring consistency.

This project demonstrates how “database systems” can solve real-world problems in the medical field and highlights the role of data integrity and structured design in supporting critical healthcare processes.

# **ER-Diagram**

A diagram of a medical organization

AI-generated content may be incorrect.

## **Entities and Relationships Description**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Description** | **Attributes** |
| Doctor | Represents a medical professional who issues prescriptions. | Doctor ID, Name, Qualification, Specialization |
| Patient | Represents a person receiving medical treatment and prescriptions. | Patient ID, Name |
| Disease | Represents a medical condition that medicines are used to treat | Disease ID, Disease Name, Disease Type |
| Medicine | Represents a drug used in  the treatment of diseases. | Medicine ID, Medicine Name, Manufacture Date, Expiry Date, Price, Dosage |
| Prescription | Represents a record of medicines issued to a patient by a doctor. | Prescription ID, Doctor ID, Patient ID, Issue Date |
| Bill | Represents the final bill for a prescription, including tax and discount. | Bill ID, Prescription ID, Tax, Discount, Total |
| Prescription \_ Medicine | Represents the quantity of each medicine included in a specific prescription. | Prescription ID, Medicine ID, Quantity |
| Medicine \_ Disease | Represents which medicines are used for which diseases. | Medicine ID, Disease ID |

## **Relationships**

|  |  |  |
| --- | --- | --- |
| **Cardinality type** | **Between** | **Relationship** |
| One-to-many | * Doctor – Prescription * Patient – Prescription | * Doctor writes Prescription * Patient receives Prescription |
| Many-to-Many | * Prescription – Medicine (via join table) * Medicine – Disease (via join table) | * Prescription includes Medicine * Medicine treats Disease |
| One-to-One | * Prescription – Bill | * Prescription has a Bill |

# **Relational Schema**

**Doctor**



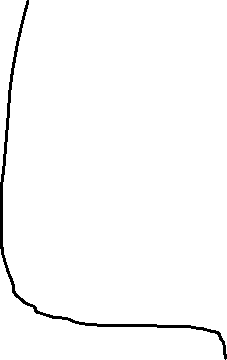
|  |  |  |  |
| --- | --- | --- | --- |
| Doctor ID | Name | Qualification | specialization |



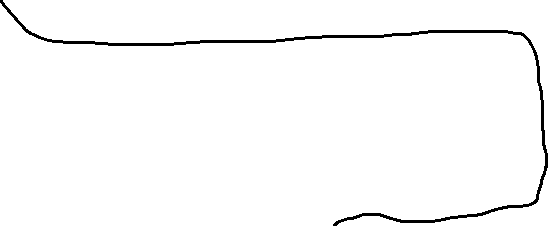
**Patient**

|  |  |  |
| --- | --- | --- |
| Patient Id | Name | Disease ID |

**Disease**



|  |  |  |
| --- | --- | --- |
| Disease ID | Disease Name | Disease Type |



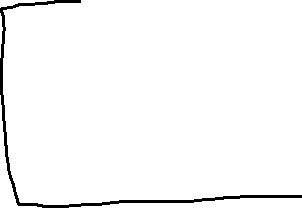
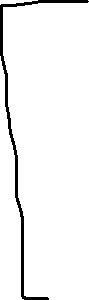
**Medicine**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Medicine ID | Medicine Name | Manufacture Date | Expiry Date | Price | Dosage |

**Prescription**



|  |  |  |  |
| --- | --- | --- | --- |
| Prescription ID | Doctor ID | Patient ID | Issue Date |



**Prescription \_ Medicine**

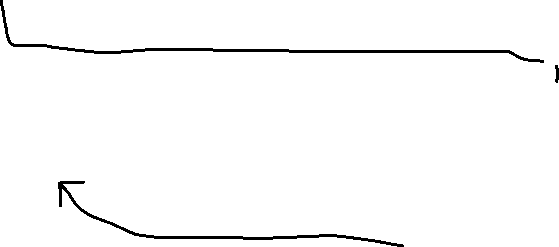


|  |  |  |
| --- | --- | --- |
| Prescription ID | Medicine ID | Quantity |

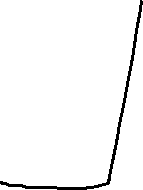


**Medicine \_ Disease**

|  |  |
| --- | --- |
| Medicine ID | Disease ID |



**Bill**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bill ID | Prescription ID | Tax | Discount | Total |

# **SQL IMPLEMENTATION**

## **Create table statements**

|  |
| --- |
| CREATE DATABASE hospital\_db;  USE hospital\_db;  -- Doctor table  CREATE TABLE Doctor (  doctor\_id INT PRIMARY KEY,  doctor\_name VARCHAR(100) NOT NULL,  qualification VARCHAR(100),  specialization VARCHAR(100)  );  -- Disease table  CREATE TABLE Disease (  disease\_id INT PRIMARY KEY,  disease\_name VARCHAR(100) NOT NULL,  disease\_type VARCHAR(50) CHECK (disease\_type IN ('infectious', 'deficiency', 'genetic hereditary', 'non-genetic hereditary'))  );  -- Medicine table  CREATE TABLE Medicine (  medicine\_id INT PRIMARY KEY,  medicine\_name VARCHAR(100) NOT NULL,  manufacture\_date DATE,  expiry\_date DATE,  price DECIMAL(10,2),  dosage VARCHAR(50)  );  -- Prescription table  CREATE TABLE Prescription (  prescription\_id INT PRIMARY KEY,  doctor\_id INT,  patient\_name VARCHAR(100) NOT NULL,  issue\_date DATE,  FOREIGN KEY (doctor\_id) REFERENCES Doctor(doctor\_id)  );  -- Prescription\_Medicine table  CREATE TABLE Prescription\_Medicine (  prescription\_id INT,  medicine\_id INT,  quantity INT,  PRIMARY KEY (prescription\_id, medicine\_id),  FOREIGN KEY (prescription\_id) REFERENCES Prescription(prescription\_id),  FOREIGN KEY (medicine\_id) REFERENCES Medicine(medicine\_id)  );  -- Medicine\_Disease table  CREATE TABLE Medicine\_Disease (  medicine\_id INT,  disease\_id INT,  PRIMARY KEY (medicine\_id, disease\_id),  FOREIGN KEY (medicine\_id) REFERENCES Medicine(medicine\_id),  FOREIGN KEY (disease\_id) REFERENCES Disease(disease\_id)  );  -- Bill table  CREATE TABLE Bill (  bill\_id INT PRIMARY KEY,  prescription\_id INT,  tax DECIMAL(10,2),  discount DECIMAL(10,2),  total DECIMAL(10,2),  FOREIGN KEY (prescription\_id) REFERENCES Prescription(prescription\_id)  ); |

## **Insert Sample Data**

|  |
| --- |
| -- Insert sample doctors  INSERT INTO Doctor (doctor\_id, doctor\_name, qualification, specialization)  VALUES  (1, 'Dr. Sarah Ahmed', 'MBBS, MD', 'Cardiology'),  (2, 'Dr. Omar Nabil', 'MBBS, MD', 'Neurology'),  (3, 'Dr. Hana Youssef', 'MBBS', 'General Medicine');  -- Insert sample diseases  INSERT INTO Disease (disease\_id, disease\_name, disease\_type)  VALUES  (1, 'Anemia', 'deficiency'),  (2, 'COVID-19', 'infectious'),  (3, 'Hemophilia', 'genetic hereditary');  -- Insert sample medicines  INSERT INTO Medicine (medicine\_id, medicine\_name, manufacture\_date, expiry\_date, price, dosage)  VALUES  (101, 'Paracetamol', '2024-01-01', '2026-01-01', 2.50, '500mg'),  (102, 'Vitamin B12', '2023-12-01', '2025-12-01', 3.00, '1000mcg'),  (103, 'Remdesivir', '2024-02-15', '2025-08-15', 25.00, '100mg');  -- Insert sample prescriptions  INSERT INTO Prescription (prescription\_id, doctor\_id, patient\_name, issue\_date)  VALUES  (1001, 1, 'Amina Khaled', '2024-05-10'),  (1002, 2, 'Mohamed Fathy', '2024-05-15');  -- Insert medicines used in prescriptions  INSERT INTO Prescription\_Medicine (prescription\_id, medicine\_id, quantity)  VALUES  (1001, 101, 2),  (1001, 102, 1),  (1002, 103, 1);  -- Link medicines to diseases  INSERT INTO Medicine\_Disease (medicine\_id, disease\_id)  VALUES  (101, 1),  (102, 1),  (103, 2);  -- Insert blank bills (trigger will update total)  INSERT INTO Bill (bill\_id, prescription\_id, tax, discount, total)  VALUES  (1, 1001, 1.00, 0.50, 0.00),  (2, 1002, 2.00, 0.00, 0.00); |

## **Trigger for bill calculation**

|  |
| --- |
| DELIMITER //  CREATE TRIGGER Calculate\_Bill\_Total  AFTER INSERT ON Prescription\_Medicine  FOR EACH ROW  BEGIN  DECLARE total\_price DECIMAL(10,2);  -- Compute the total price from the Prescription\_Medicine table  SELECT SUM(Medicine.Price \* Prescription\_Medicine.Quantity)  INTO total\_price  FROM Prescription\_Medicine  JOIN Medicine ON Prescription\_Medicine.MedicineID = Medicine.MedicineID  WHERE Prescription\_Medicine.PrescriptionID = NEW.PrescriptionID;  -- Update the Bill table with the calculated total  UPDATE Bill  SET Total = total\_price + IFNULL(Tax, 0) - IFNULL(Discount, 0)  WHERE PrescriptionID = NEW.PrescriptionID;  END;  //  DELIMITER ; |

# **Queries and sample output**

|  |
| --- |
| SELECT Name  FROM Doctor  WHERE Specialization LIKE '%heart%'; |

|  |
| --- |
| SELECT DiseaseName  FROM Disease  WHERE DiseaseType = 'deficiency'; |

|  |
| --- |
| SELECT M.MedicineName, SUM(PM.Quantity) AS TotalSold  FROM Prescription P  JOIN Prescription\_Medicine PM ON P.PrescriptionID = PM.PrescriptionID  JOIN Medicine M ON PM.MedicineID = M.MedicineID  WHERE YEAR(P.IssueDate) = 2023  GROUP BY M.MedicineID  ORDER BY TotalSold DESC  LIMIT 1; |

# **6.Conclusion**

In this project, we developed a Medicine Inventory Management System that efficiently tracks medicine stock levels, manages prescriptions, and automatically calculates billing through database triggers. The system ensures that medicine quantities are accurately updated, and billing amounts reflect the prescribed medicines and their quantities.

During the implementation, we encountered some challenges such as:

* Designing triggers with correct syntax and logic compatible with MySQL.
* Ensuring the integrity of inventory updates when new prescriptions are added.
* Handling calculations of total bills while incorporating taxes and discounts correctly.
* Managing the relationships between multiple tables (Medicines, Prescriptions, Bills) to maintain consistent data.

To address these, we performed extensive testing and refined our SQL queries and trigger logic to ensure smooth and accurate operations.

Suggestions for Future Improvements:

* Implementing advanced stock alerts for low inventory levels to prevent stockouts.
* Adding features for automatic expiry date tracking and notifications for medicines nearing expiration.
* Developing a more user-friendly interface to facilitate easy management of inventory and billing.
* Introducing detailed reporting tools to analyze sales, inventory turnover, and prescription trends.
* Enhancing billing calculations to support variable discounts, promotions, and insurance processing.