Use DBML to define your database structure

Docs: https:dbml.dbdiagram.io/docs

Table users {

id INT [primary key, increment]

username VARCHAR(50) [not null, unique]

password VARCHAR(255) [not null]

role VARCHAR(50) [not null]

permissions TEXT

}

Table medicines {

id INT [primary key, increment]

name VARCHAR(255) [not null]

inventory\_price DECIMAL(10, 2) [not null]

sale\_price DECIMAL(10, 2) [not null]

stock INT [not null]

prescription\_needed BOOLEAN [default: false]

created\_at TIMESTAMP [default: `CURRENT\_TIMESTAMP`]

}

Table sales {

id INT [primary key, increment]

medicine\_id INT [not null]

quantity INT [not null]

unit\_price DECIMAL(10, 2) [not null]

total\_price DECIMAL(10, 2) [not null]

profit DECIMAL(10, 2) [not null]

customer\_name VARCHAR(255) [not null]

sale\_date TIMESTAMP [default: `CURRENT\_TIMESTAMP`]

}

Table expiration\_management {

id INT [primary key, increment]

medicine\_id INT [not null]

batch\_number VARCHAR(255) [not null]

expiration\_date DATE [not null]

quantity INT [not null]

}

Table suppliers {

id INT [primary key, increment]

name VARCHAR(255) [not null]

contact\_person VARCHAR(255) [not null]

email VARCHAR(255) [not null]

phone VARCHAR(20) [not null]

address TEXT [not null]

}

Table prescriptions {

id INT [primary key, increment]

prescription\_id VARCHAR(255) [not null]

patient\_name VARCHAR(255) [not null]

doctor\_name VARCHAR(255) [not null]

prescription\_date DATE [not null]

status ENUM('Pending', 'Filled') [default: 'Pending']

notes TEXT

created\_at TIMESTAMP [default: `CURRENT\_TIMESTAMP`]

}

Table prescribed\_medicines {

id INT [primary key, increment]

prescription\_id INT [not null]

medicine\_id INT [not null]

quantity INT [not null]

dosage VARCHAR(255) [not null]

}

Relationships

Ref: sales.medicine\_id > medicines.id

Ref: expiration\_management.medicine\_id > medicines.id

Ref: prescribed\_medicines.prescription\_id > prescriptions.id

Ref: prescribed\_medicines.medicine\_id > medicines.id

// Queries for Different Functionalities

// Update Stock

UPDATE medicines SET stock = stock + 10 WHERE id = 1;

// Check Stock

SELECT name, stock FROM medicines;

// Inventory Report

SELECT

m.name,

m.inventory\_price,

m.sale\_price,

m.stock,

(m.inventory\_price \* m.stock) AS total\_value,

m.prescription\_needed

FROM medicines m;

// Sales Records

SELECT

s.sale\_date,

m.name AS medicine\_name,

s.customer\_name,

s.quantity,

s.total\_price,

s.profit

FROM sales s

JOIN medicines m ON s.medicine\_id = m.id;

// Top Selling Analysis

SELECT

m.name AS medicine\_name,

SUM(s.quantity) AS total\_quantity\_sold,

SUM(s.total\_price) AS total\_revenue,

SUM(s.profit) AS total\_profit

FROM sales s

JOIN medicines m ON s.medicine\_id = m.id

GROUP BY m.name

ORDER BY total\_quantity\_sold DESC;

// Statistics

SELECT

(SELECT COUNT(\*) FROM medicines) AS total\_medicines,

(SELECT SUM(stock) FROM medicines) AS total\_stock,

(SELECT COUNT(\*) FROM medicines WHERE stock < 50) AS low\_stock\_items,

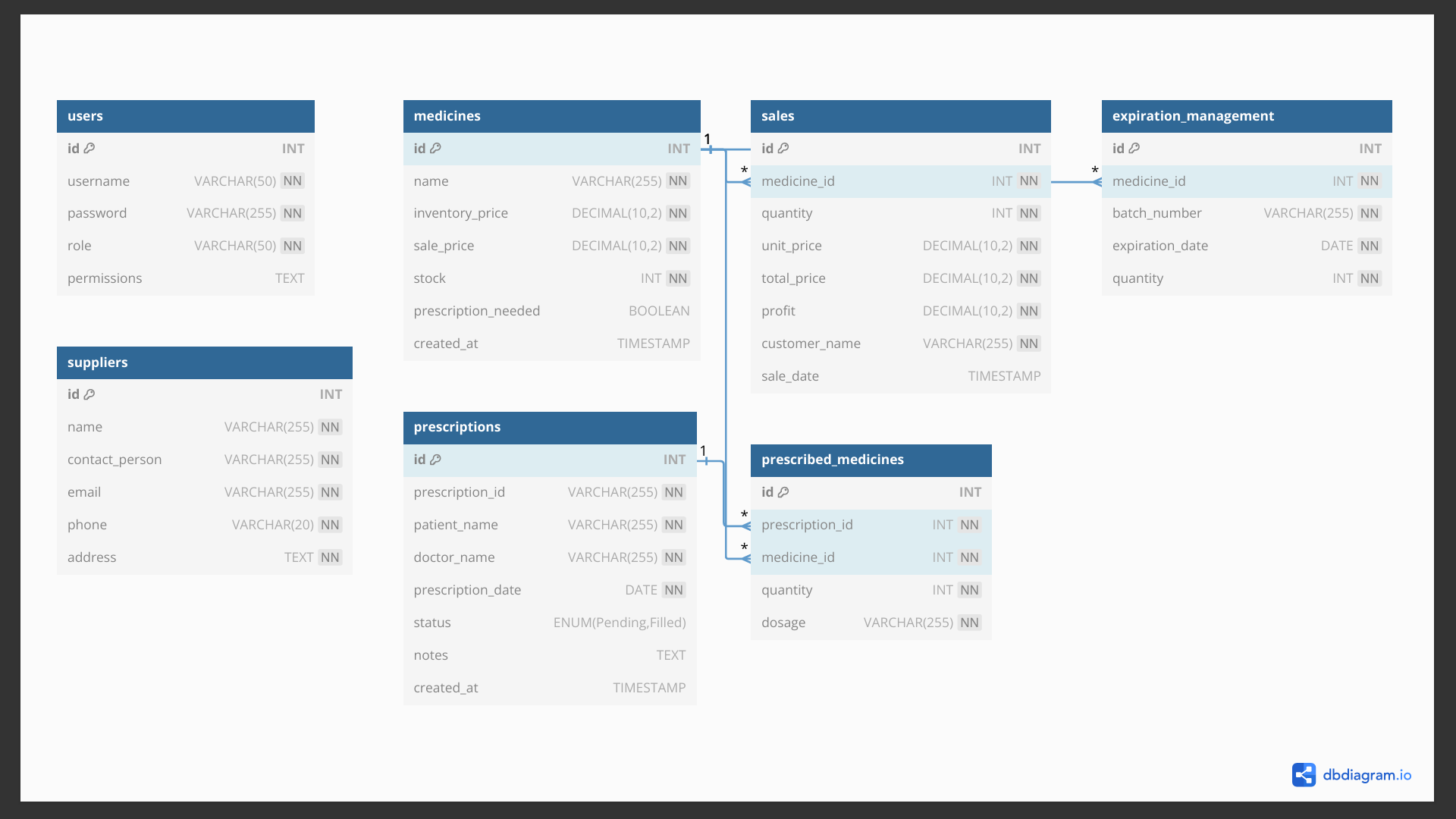
(SELECT SUM(inventory\_price \* stock) FROM medicines) AS total\_stock\_value,

(SELECT COUNT(\*) FROM sales) AS total\_sales\_count,

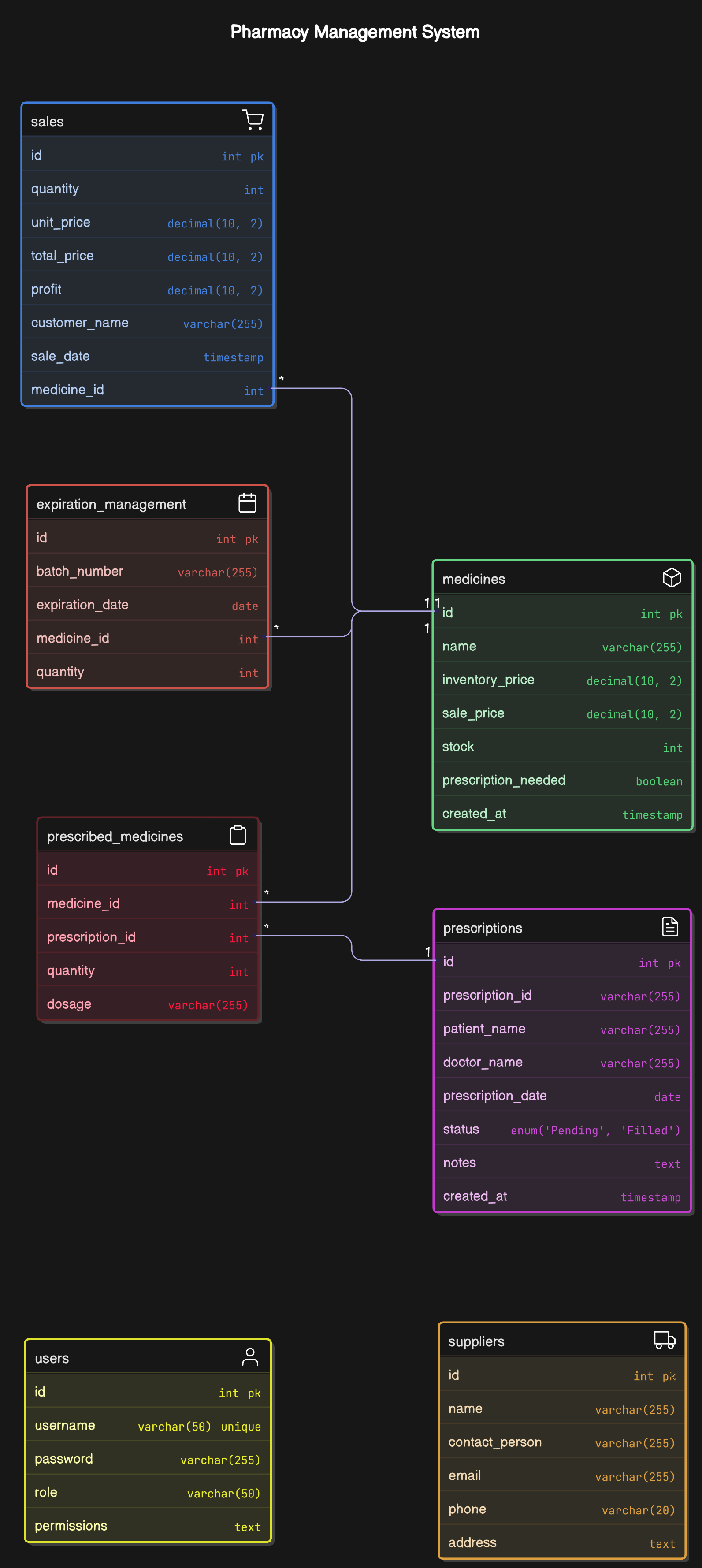
(SELECT SUM(total\_price) FROM sales) AS total\_sales\_revenue,

(SELECT SUM(profit) FROM sales) AS total\_sales\_profit,

(SELECT (SUM(profit) / SUM(total\_price)) \* 100 FROM sales) AS profit\_margin;

**SCHEMA DIAGRAM:**  


**ER MODEL:**



**SEQUENCE DIAGRAM CODE:**  
title Database Operations Sequence Diagram

participant User

participant UsersTable

participant MedicinesTable

participant SalesTable

participant ExpirationManagementTable

participant SuppliersTable

participant PrescriptionsTable

participant PrescribedMedicinesTable

== User Registration ==

User->UsersTable: Register(username, password, role, permissions)

UsersTable-->User: User registered successfully

== Medicine Management ==

User->MedicinesTable: AddMedicine(name, inventory\_price, sale\_price, stock, prescription\_needed)

MedicinesTable-->User: Medicine added successfully

User->MedicinesTable: UpdateStock(medicine\_id, quantity)

MedicinesTable-->User: Stock updated successfully

User->MedicinesTable: CheckStock()

MedicinesTable-->User: Return stock information

== Sales Management ==

User->SalesTable: RecordSale(medicine\_id, quantity, unit\_price, total\_price, profit, customer\_name)

SalesTable->MedicinesTable: UpdateStock(medicine\_id, -quantity)

MedicinesTable-->SalesTable: Stock updated

SalesTable-->User: Sale recorded successfully

== Expiration Management ==

User->ExpirationManagementTable: AddBatch(medicine\_id, batch\_number, expiration\_date, quantity)

ExpirationManagementTable-->User: Batch added successfully

== Supplier Management ==

User->SuppliersTable: AddSupplier(name, contact\_person, email, phone, address)

SuppliersTable-->User: Supplier added successfully

== Prescription Management ==

User->PrescriptionsTable: AddPrescription(prescription\_id, patient\_name, doctor\_name, prescription\_date, status, notes)

PrescriptionsTable-->User: Prescription added successfully

User->PrescribedMedicinesTable: AddPrescribedMedicine(prescription\_id, medicine\_id, quantity, dosage)

PrescribedMedicinesTable-->User: Prescribed medicine added successfully

== Reports and Analytics ==

User->MedicinesTable: GenerateInventoryReport()

MedicinesTable-->User: Return inventory report

User->SalesTable: GenerateSalesRecords()

SalesTable-->User: Return sales records

User->SalesTable: GenerateTopSellingAnalysis()

SalesTable-->User: Return top selling analysis

User->SalesTable: GenerateStatistics()

SalesTable-->User: Return statistics

A black and white image of a diagram

AI-generated content may be incorrect.

**ACTIVITY DIAGRAM:**

A black and white screen

AI-generated content may be incorrect.