School of Computing and Design
COMPUTER SCIENCE
California State University
MONTEREY BAY

Pharmacial Database

Dawn Petersen, Nima Mahanloo, Jerry Kim Do

Professor Wisneski

CST 363

May 30, 2023

Table of Contents

Project Overview……Page 1

ER Diagram……Page 2

Entities, Attributes and Relationships……Page 3-5

Relational Schema…… Page 6-10

5 Sample Management Questions ……...…… Page 11

Appendix I: Program Screenshots…… Page 12-16

Appendix II: Screenshots for Handling User Input Exceptions…… Page 17-19

Conclusion……..Page 20

Project Overview

The pharmacy management system is a database application system, which was developed in MySQL and connected to a front end to manage data collection, entry, and queries.

In this project, we will design and implement a normalized database that allows pharmacies to manage their inventory and prescriptions, and for doctors to prescribe drugs to patients. The system will also enable pharmacies to manage their contracts with pharmaceutical companies.

We will capture a number of entities in this database. First, the patient entity which will be identified by a Social Security Number(SSN), and has a name, age and address. Second, a Doctor, who will also be identified by a Social Security Number(SSN), and has a name, specialty, and years of experience. Next, the Pharmaceutical Company, which will be identified by name, and has a phone number. Next, drugs, which will have a generic name and may have a unique trade name. Then Pharmacy, which is identified by name, address, and phone number. Then Prescription, which has a unique number(RX ID) and can only belong to only one drug, and patient, and is written by only one doctor. Next Contract, which connects the pharmacy and the pharmaceutical company, has a start date and an end date. Lastly contract terms, which contains the supervisor's details, managing a particular contract.

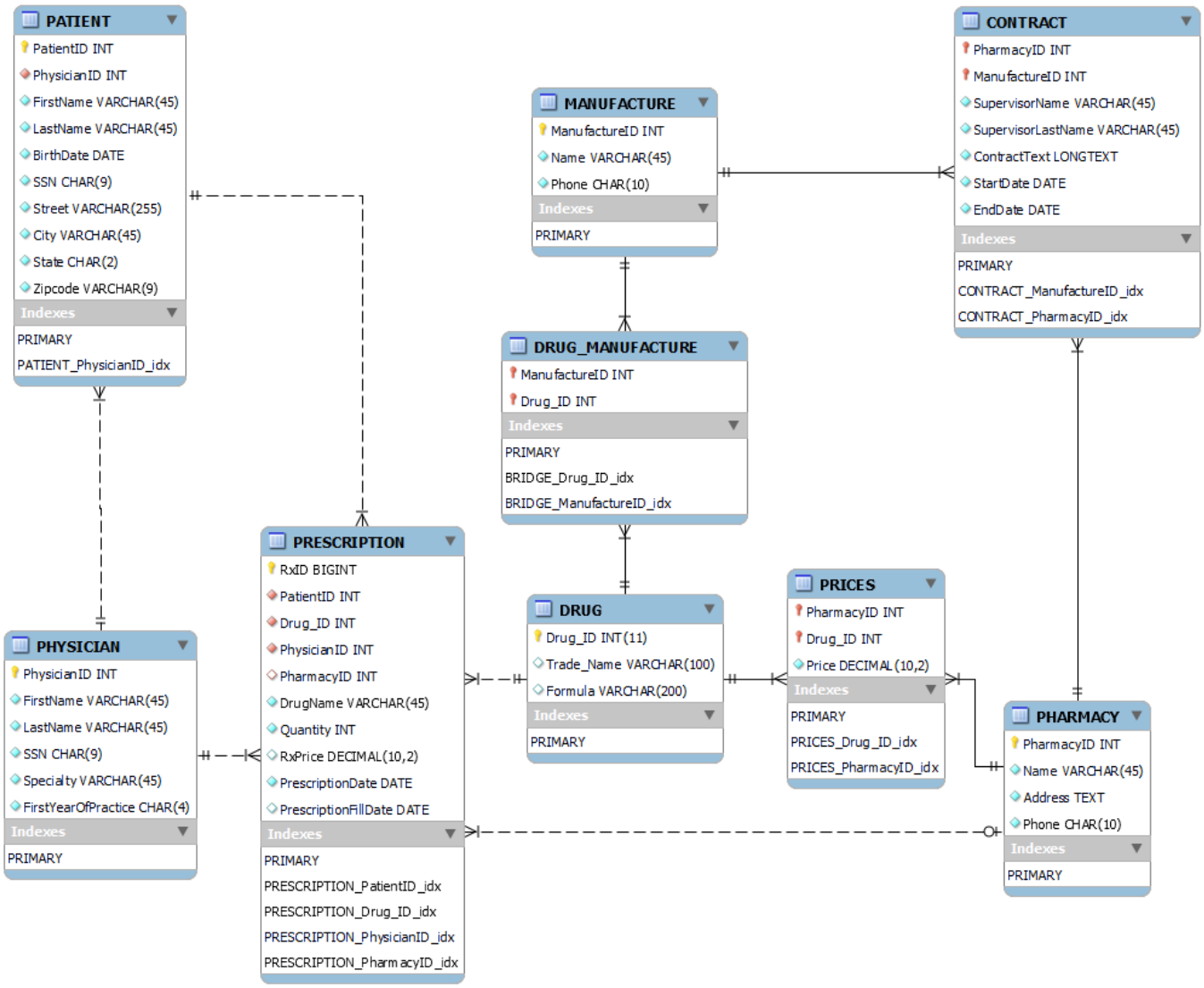
Apart from the Price and the Contract tables, every other table in this database will have a unique identifier which is the primary key. This key is unique to each table. And a foreign key for reference purposes.

There are several assumptions we made while developing the schema. To start, a patient can have only one primary physician. Any physician can write a prescription for any patient. Each pharmacy displays and sells more than one drug and has a price for each. Doctors prescribe drugs for patients using a prescription and a prescription contains a date and an associated quantity. If the prescription has a trade name, it is for a specific drug from a specific company. If the prescription is for a generic name, any drug with that formula name from any pharmaceutical company can be used. When a prescription is filled, the system tracks the pharmacy that is stored at the PrescriptionFillDate. If a prescription is for a generic drug, the system tracks which pharmaceutical company supplied the drug. Our last assumption is that Pharmaceutical companies can contract with several pharmacies, and vice versa.

Our system functionality allows patients to be registered, and for doctors to give prescriptions. Pharmacies can also be registered and manage their inventory of drugs, prices, and prescriptions. Pharmaceutical companies can also be registered, and they can manage their contracts with pharmacies. We will also provide a search function for finding prescriptions, drugs, and pharmacies. And generate reports on sales, inventory, prescriptions, and contracts, which also provide useful statistics for pharmacy managers to improve their business.

Our overall goal is to have a working and usable pharmacy management system that will allow pharmacies to manage their inventory of drugs and prescriptions, and for doctors to prescribe drugs to patients. Our goal is also to enable pharmacies to manage their contracts with pharmaceutical companies. The system will be developed using MySQL as the database engine, and will provide fast search and retrieval of data.

ER Diagram

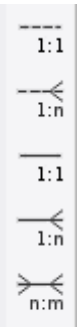


Normalization

As the diagram and database show and prove that clearly, we avoided unnecessary duplication and repetition of data objects and columns in this project. We used only minimum necessity columns for this project, and each column holds only a single data value. We divided and distributed columns into separated independent compact units as necessary tables that each contains only its own direct dependent data columns without any partial dependency. Each table has its primary key, and there are no multivalued dependencies on a primary Key. Tables have a normal relationship with each other as necessary through the necessary foreign keys. We also designed and managed prime and non-prime attributes properly. For each functional dependency X->Y, X is a super key of each table, and Y is a prime attribute of the table. Also, no composite key has any cyclic dependencies. Therefore, we are sure that our database normalized in 5nf properly.

Attributes

Relationship Connection Lines



solid lines = direct relationship (sharing primary key)
dotted lines = indirect relationship (not sharing primary key)
1:1 = one to one relationship
1:n = one to many relationship
n:m = many to many relationship

Icons

Key: Primary Key
Foreign Primary Key
Filled Diamond: NOT NULL
Not filled Diamond: CAN BE NULL
Red colored: (Part of) Foreign key
Blue lined Diamond: Simple attribute (no key)

Entities

PATIENT……...	provides the patient's identification, and location information.
PHYSICIAN…….	provides the physician's identification, specialty, and first year of practice.
PRESCRIPTION……..	holds information about PATIENT, DRUG, PHYSICIAN, and PHARMACY, and also includes the prescription date and when it should be refilled.
PHARMACY……	provides the location and contact information for the pharmacy business.
CONTRACT…….	allows pharmacies to sell manufactured drugs, including agreement and supervisor name
MANUFACTURE……	provides the name and contact information for the manufacturing business.
DRUG MANUFACTURE……	specifies which company manufactures a certain drug product
PRICES…….	is necessary for the drug's price being prescribed to patients.
DRUGS…….	provides information about the drug's names, dosage, quantity, and its manufacturer.

Relationships

relationship	type	description
PHYSICIAN - PATIENT	1:n	a single physician may be medically responsible to multiple patients at a time.
PHYSICIAN - PRESCRIPTION	1:n	a single physician may assign multiple prescriptions to a single patient.
PATIENT - PRESCRIPTION	1:n	a single patient may have multiple prescriptions assigned by the physician.
PHARMACY - PRESCRIPTION	1:n	a pharmacy may produce multiple prescriptions to patients.
PHARMACY - CONTRACT	1:n	a pharmacy would have multiple contract agreements with a manufacturer.
PHARMACY - PRICE	1:n	a pharmacy would have multiple prices for multiple drugs.
MANUFACTURE - CONTRACT	1:n	a manufacturer may produce multiple contracts for pharmacies.
MANUFACTURE - DRUG_MANUFACTURE	1:n	a manufacturer may produce various drug brands
DRUG - DRUG_MANUFACTURE	1:n	a specific drug may have various drug brands manufactured after it
DRUGS - PRESCRIPTION	1:n	a specific drug may be prescribed multiple times to various patients
DRUGS - PRICES	1:n	a drug with specified quantity and dosage has various prices among pharmacies.

Relational Schema

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTIO
N';

-- -----------------------------------------------------
-- Schema PHARMACIES
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `PHARMACIES` DEFAULT CHARACTER SET utf8 ;
USE `PHARMACIES` ;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`PHYSICIAN`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PHYSICIAN` (
  `PhysicianID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(45) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `SSN` CHAR(9) NOT NULL,
  `Specialty` VARCHAR(45) NOT NULL,
  `FirstYearOfPractice` CHAR(4) NOT NULL,
  PRIMARY KEY (`PhysicianID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`PATIENT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PATIENT` (
  `PatientID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `PhysicianID` INT UNSIGNED NOT NULL,
  `FirstName` VARCHAR(45) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `BirthDate` DATE NOT NULL,
  `SSN` CHAR(9) NOT NULL,
  `Street` VARCHAR(255) NOT NULL,
  `City` VARCHAR(45) NOT NULL,
  `State` CHAR(2) NOT NULL,
  `Zipcode` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`PatientID`),
  INDEX `PATIENT_PhysicianID_idx` (`PhysicianID` ASC) VISIBLE,
  CONSTRAINT `PATIENT_PhysicianID_idx`
    FOREIGN KEY (`PhysicianID`)
    REFERENCES `PHARMACIES`.`PHYSICIAN` (`PhysicianID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`PHARMACY`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PHARMACY` (
  `PharmacyID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(45) NOT NULL,
  `Address` TEXT NOT NULL,
  `Phone` CHAR(10) NOT NULL,
  PRIMARY KEY (`PharmacyID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`MANUFACTURE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`MANUFACTURE` (
  `ManufactureID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
  `Name` VARCHAR(45) NOT NULL,
  `Phone` CHAR(10) NOT NULL,
  PRIMARY KEY (`ManufactureID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `PHARMACIES`.`DRUG`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`DRUG` (
  `Drug_ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
  `Trade_Name` VARCHAR(100) NULL DEFAULT NULL,
  `Formula` VARCHAR(200) NULL DEFAULT NULL,
  PRIMARY KEY (`Drug_ID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`PRESCRIPTION`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PRESCRIPTION` (
  `RxID` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `PatientID` INT UNSIGNED NOT NULL,
  `Drug_ID` INT UNSIGNED NOT NULL,
  `PhysicianID` INT UNSIGNED NOT NULL,
  `PharmacyID` INT UNSIGNED NULL,
  `DrugName` VARCHAR(45) NOT NULL,
  `Quantity` INT UNSIGNED NOT NULL,
  `RxPrice` DECIMAL(10,2) UNSIGNED NULL,
  `PrescriptionDate` DATE NOT NULL DEFAULT (CURRENT_DATE()),
  `PrescriptionFillDate` DATE NULL,
  PRIMARY KEY (`RxID`),
  INDEX `PRESCRIPTION_PatientID_idx` (`PatientID` ASC) VISIBLE,
  INDEX `PRESCRIPTION_Drug_ID_idx` (`Drug_ID` ASC) VISIBLE,
  INDEX `PRESCRIPTION_PhysicianID_idx` (`PhysicianID` ASC) VISIBLE,
  INDEX `PRESCRIPTION_PharmacyID_idx` (`PharmacyID` ASC) VISIBLE,
  CONSTRAINT `PRESCRIPTION_PatientID_idx`
    FOREIGN KEY (`PatientID`)
    REFERENCES `PHARMACIES`.`PATIENT` (`PatientID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `PRESCRIPTION_Drug_ID_idx`
    FOREIGN KEY (`Drug_ID`)
    REFERENCES `PHARMACIES`.`DRUG` (`Drug_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `PRESCRIPTION_PhysicianID_idx`
    FOREIGN KEY (`PhysicianID`)
    REFERENCES `PHARMACIES`.`PHYSICIAN` (`PhysicianID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `PRESCRIPTION_PharmacyID_idx`
    FOREIGN KEY (`PharmacyID`)
    REFERENCES `PHARMACIES`.`PHARMACY` (`PharmacyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`PRICES`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PRICES` (
  `PharmacyID` INT UNSIGNED NOT NULL,
  `Drug_ID` INT UNSIGNED NOT NULL,
  `Price` DECIMAL(10,2) UNSIGNED NOT NULL,
  PRIMARY KEY (`PharmacyID`, `Drug_ID`),
  INDEX `PRICES_Drug_ID_idx` (`Drug_ID` ASC) VISIBLE,
  INDEX `PRICES_PharmacyID_idx` (`PharmacyID` ASC) VISIBLE,
  CONSTRAINT `PRICES_PharmacyID_idx`
    FOREIGN KEY (`PharmacyID`)
    REFERENCES `PHARMACIES`.`PHARMACY` (`PharmacyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `PRICES_Drug_ID_idx`
    FOREIGN KEY (`Drug_ID`)
    REFERENCES `PHARMACIES`.`DRUG` (`Drug_ID`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`CONTRACT`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`CONTRACT` (
  `PharmacyID` INT UNSIGNED NOT NULL,
  `ManufactureID` INT UNSIGNED NOT NULL,
  `SupervisorName` VARCHAR(45) NOT NULL,
  `SupervisorLastName` VARCHAR(45) NOT NULL,
  `ContractText` LONGTEXT NOT NULL,
  `StartDate` DATE NOT NULL,
  `EndDate` DATE NOT NULL,
  PRIMARY KEY (`PharmacyID`, `ManufactureID`),
  INDEX `CONTRACT_ManufactureID_idx` (`ManufactureID` ASC) VISIBLE,
  INDEX `CONTRACT_PharmacyID_idx` (`PharmacyID` ASC) VISIBLE,
  CONSTRAINT `PharmacyID`
    FOREIGN KEY (`PharmacyID`)
    REFERENCES `PHARMACIES`.`PHARMACY` (`PharmacyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ManufactureID`
    FOREIGN KEY (`ManufactureID`)
    REFERENCES `PHARMACIES`.`MANUFACTURE` (`ManufactureID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `PHARMACIES`.`DRUG_MANUFACTURE`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`DRUG_MANUFACTURE` (
  `ManufactureID` INT UNSIGNED NOT NULL,
  `Drug_ID` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`ManufactureID`, `Drug_ID`),
  INDEX `BRIDGE_Drug_ID_idx` (`Drug_ID` ASC) VISIBLE,
  INDEX `BRIDGE_ManufactureID_idx` (`ManufactureID` ASC) VISIBLE,
  CONSTRAINT `BRIDGE_ManufactureID`
    FOREIGN KEY (`ManufactureID`)
    REFERENCES `PHARMACIES`.`MANUFACTURE` (`ManufactureID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `BRIDGE_Drug_ID`
    FOREIGN KEY (`Drug_ID`)
    REFERENCES `PHARMACIES`.`DRUG` (`Drug_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `PHARMACIES` ;

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`PATIENT_INFO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PATIENT_INFO` (`PatientID` INT, `'Patient'` INT, `'Date of Birth'` INT, `'Social
Security Number'` INT, `'Address'` INT, `'Primary Care'` INT, `'Primary Care SSN'` INT);

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`PRESCRIPTION_INFO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PRESCRIPTION_INFO` (`RxID` INT, `'Prescribed at'` INT, `'Medication'` INT,
`'Manufacture'` INT, `'Quantity'` INT, `'Fill Date'` INT, `'Prescribed for'` INT, `'Patient SSN'` INT, `'Prescriber'` INT,
`'Physician SSN'` INT, `'Specialty'` INT, `'Pharmacy'` INT, `'Pharmacy Address'` INT, `'Pharmacy Phone'` INT, `'Price'` INT);

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`DRUG_INFO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`DRUG_INFO` (`Drug_ID` INT, `'Trade Name'` INT, `'Generic Name'` INT, `'Manufacture'`
INT);

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`CONTRACT_INFO`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`CONTRACT_INFO` (`'Contract Between'` INT, `'From'` INT, `'To'` INT, `'Manufacture
Phone'` INT, `'Pharmacy Supervisor'` INT, `'Pharmacy Address'` INT, `'Pharmacy Phone'` INT, `'Text'` INT);

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`PHYSICIAN_INFO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PHYSICIAN_INFO` (`PhysicianID` INT, `'Physician Name'` INT, `SSN` INT, `Specialty`
INT, `'First Year of Practice'` INT);

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`PHARMACY_INFO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`PHARMACY_INFO` (`PharmacyID` INT, `'Store Name'` INT, `'Store Address'` INT, `Phone`
INT);

-- -----------------------------------------------------
-- Placeholder table for view `PHARMACIES`.`MANUFACTURE_INFO`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `PHARMACIES`.`MANUFACTURE_INFO` (`ManufactureID` INT, `'Manufacture Name'` INT, `Phone` INT);

-- -----------------------------------------------------
-- View `PHARMACIES`.`PATIENT_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`PATIENT_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `PHARMACIES`.`PATIENT_INFO` AS
         SELECT P.PatientID, CONCAT(P.FirstName, ' ', P.LastName) AS 'Patient',
    P.BirthDate AS 'Date of Birth', P.SSN AS 'Social Security Number',
    CONCAT(P.Street, ' ', P.City, ' ', P.State, ' ', P.Zipcode) AS 'Address',
         CONCAT(D.FirstName, ' ', D.LastName) AS 'Primary Care',  D.SSN AS 'Primary Care SSN'
    FROM PATIENT P JOIN PHYSICIAN D
         ON P.PhysicianID = D.PhysicianID
         ORDER BY P.FirstName;

-- -----------------------------------------------------
-- View `PHARMACIES`.`PRESCRIPTION_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`PRESCRIPTION_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `PHARMACIES`.`PRESCRIPTION_INFO` AS
         SELECT S.RxID, S.PrescriptionDate AS 'Prescribed at',
         S.DrugName AS 'Medication', C.Name AS 'Manufacture',
    S.Quantity AS 'Quantity', S.PrescriptionFillDate AS 'Fill Date',
         CONCAT(P.FirstName, ' ', P.LastName) AS 'Prescribed for', P.SSN AS 'Patient SSN',
    CONCAT(D.FirstName, ' ', D.LastName) AS 'Prescriber', D.SSN AS 'Physician SSN',
         D.Specialty AS 'Specialty',
         F.Name AS 'Pharmacy', F.Address AS 'Pharmacy Address', F.Phone AS 'Pharmacy Phone',
         S.RxPrice AS 'Price'
         FROM PRESCRIPTION S, DRUG M, MANUFACTURE C, DRUG_MANUFACTURE MC, PATIENT P, PHYSICIAN D, PHARMACY F, PRICES R
         WHERE S.PatientID = P.PatientID AND S.Drug_ID = M.Drug_ID AND M.Drug_ID = MC.Drug_ID AND MC.ManufactureID =
C.ManufactureID
         AND S.PhysicianID = D.PhysicianID AND S.PharmacyID = F.PharmacyID
         AND R.Drug_ID = M.Drug_ID AND R.PharmacyID = F.PharmacyID
         ORDER BY S.PrescriptionDate DESC;

-- -----------------------------------------------------
-- View `PHARMACIES`.`DRUG_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`DRUG_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `PHARMACIES`.`DRUG_INFO` AS
         SELECT D.Drug_ID, D.Trade_Name AS 'Trade Name', D.Formula AS 'Generic Name',
         M.Name AS 'Manufacture'
         FROM DRUG D, DRUG_MANUFACTURE DM, MANUFACTURE M
         WHERE D.Drug_ID = DM.Drug_ID AND DM.ManufactureID = M.ManufactureID
         ORDER BY D.Trade_Name;

-- -----------------------------------------------------
-- View `PHARMACIES`.`CONTRACT_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`CONTRACT_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `PHARMACIES`.`CONTRACT_INFO` AS
         SELECT CONCAT(M.Name, ' - ', P.Name) AS 'Contract Between', StartDate AS 'From', EndDate AS 'To',
         M.Phone AS 'Manufacture Phone',
```

```
        CONCAT(C.SupervisorName, ' ', C.SupervisorLastName) AS 'Pharmacy Supervisor',
        P.Address AS 'Pharmacy Address',  P.Phone AS 'Pharmacy Phone',
        C.ContractText AS 'Text'
        FROM CONTRACT C, PHARMACY P, MANUFACTURE M
        WHERE C.PharmacyID = P.PharmacyID AND C.ManufactureID = M.ManufactureID
        ORDER BY M.NAME;

-- -----------------------------------------------------
-- View `PHARMACIES`.`PHYSICIAN_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`PHYSICIAN_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `PHYSICIAN_INFO` AS
        SELECT PhysicianID, CONCAT(FirstName, ' ', LastName) AS 'Physician Name',
    SSN, Specialty, FirstYearOfPractice AS 'First Year of Practice'
    FROM PHYSICIAN
    ORDER BY FirstName;

-- -----------------------------------------------------
-- View `PHARMACIES`.`PHARMACY_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`PHARMACY_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `PHARMACY_INFO` AS
        SELECT PharmacyID, Name AS 'Store Name',
    Address AS 'Store Address', Phone
    FROM PHARMACY
    ORDER BY Name;

-- -----------------------------------------------------
-- View `PHARMACIES`.`MANUFACTURE_INFO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PHARMACIES`.`MANUFACTURE_INFO`;
USE `PHARMACIES`;
CREATE  OR REPLACE VIEW `MANUFACTURE_INFO` AS
        SELECT ManufactureID, Name AS 'Manufacture Name', Phone
    FROM MANUFACTURE
    ORDER BY Name;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

5 Sample Management Questions

In the following, we provided some statistical questions that may be asked by the management department, and the following SQL codes to answer the questions:

1. Display doctors in order with the most experience?

```
SELECT PhysicianID, FirstName, LastName, Specialty,
     CAST(FirstYearOfPractice AS UNSIGNED) AS InPracticeSince
     FROM PHYSICIAN ORDER BY InPracticeSince ASC;
```

2. Display drugs in order with the most prescribed?

```
SELECT D.*, COUNT(P.Drug_ID) PrescribedTimes
     FROM DRUG D JOIN PRESCRIPTION P ON D.Drug_ID = P.Drug_ID
     GROUP BY P.Drug_ID ORDER BY PrescribedTimes DESC;
```

3. Who are the oldest patients?

```
SELECT * FROM PATIENT
     WHERE BirthDate = (SELECT MIN(BirthDate) FROM PATIENT);
```

4. Doctor with the most prescriptions?

```
SELECT * FROM PHYSICIAN
     WHERE PhysicianID =
     (SELECT PhysicianID FROM PRESCRIPTION
     GROUP BY PhysicianID HAVING COUNT(PhysicianID) =
     (SELECT MAX(T.CNT) FROM
     (SELECT COUNT(*) AS CNT FROM PRESCRIPTION
     GROUP BY PhysicianID) AS T));
```

5. Doctor with the most patients?

```
SELECT P.* FROM PHYSICIAN P JOIN PATIENT T ON
     P.PhysicianID = T.PhysicianID GROUP BY T.PhysicianID
     HAVING COUNT(T.PhysicianID) = (SELECT MAX(C.CNT) FROM
     (SELECT COUNT(PhysicianID) AS CNT FROM PATIENT
     GROUP BY PhysicianID) AS C);
```

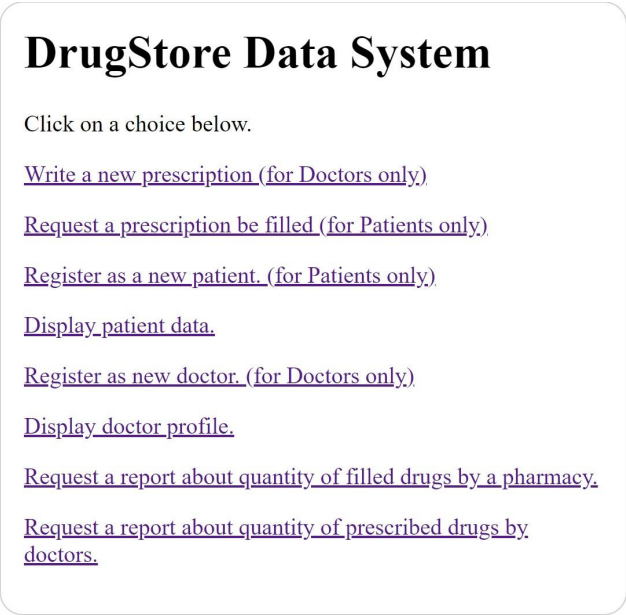
Appendix I: Program Screenshots

IMPORTANT NOTICE: In order to function the DrugStore Data System, the following assumptions must be true:

1. The database tables has populated data (can be randomly generated by running GenerateData.java)
2. The SQL Script is ran with a localhost:8080 server connection
3. The user's password for the localhost:8080 connection are inserted next to the password variables located at the top of 3 files:
    - Doctor_load_data.java
    - GenerateData.java
    - application.properties

The following screenshots will showcase every DrugStore Data System option from the main menu. Each main menu option will have the following screenshots:

- form input (left side)
- the resulting output page (right side)



Main Menu

The first 6 options are for registering data and viewing profiles. The last 2 options are for inspection purposes, typically used by a pharmacy manager or an FDA government official.

For the screenshot examples, we can assume the doctors, patients, and drugs already exist in the database when running. For prescriptions to be filled, a doctor must write a new prescription form, then a patient must use the generated Rx number to request a prescription fill.

New Prescription Form

Doctor SSN: 110272600

Doctor First Name: Cyrus

Doctor Last Name: The Great

Patient SSN: 666911666

Patient First Name: Jorge Mario

Patient Last Name: Pope Francis

Drug Name: pantoprazole

Quantity: 13

Create Prescription

Write a new prescription

Prescription created.

Rx: 101
Doctor: 110272600
First Name: Cyrus
Last Name: The Great
Patient: 666911666
First Name: Jorge Mario
Last Name: Pope Francis
Drug: pantoprazole
Quantity: 13
Pharmacy:
Name:
Address:
Phone:
Date Filled:
Cost: $

Main Menu

Written prescription output

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Rx: 101

Patient Last Name: Pope Francis

Pharmacy Name: Joseph Drugs

Pharmacy Address: 288 Xavier, Jacob, PA, 67114

Request Fill for Prescription

Request a prescription to be filled

Prescription has been filled.

Rx: 101
Doctor:
First Name: Cyrus
Last Name: The Great
Patient:
First Name: Jorge Mario
Last Name: Pope Francis
Drug: pantoprazole
Quantity: 13
Pharmacy:
Name: Joseph Drugs
Address: 288 Xavier, Jacob, PA, 67114
Phone: 0704513381
Date Filled: 2023-06-01
Cost: $

Main Menu

Filled prescription output

Register as new user

Your SSN: •••••••••

Your First Name: Jorge Mario

Your Last Name: Pope Francis

Birth Date: 12/17/1936

Street: 00120 Via del Pellegrino

City: Vatican City

State: VC

Zipcode: 13666

Primary Physician Name: Cyrus The Great

Register

Register as a new patient

Registration successful.

Patient ID:	101
First Name:	Jorge Mario
Last Name:	Pope Francis
Birthdate:	1936-12-17
Street:	00120 Via del Pellegrino
City:	Vatican City
State:	VC
Zipcode:	13666
Primary Physican:	Cyrus The Great

Edit | Main Menu

Registered patient output

Enter patient id and name

Patient ID: 101

Paticnt Last Name: Pope Francis

Get Profile

Display patient data

Patient ID:	101
First Name:	Jorge Mario
Last Name:	Pope Francis
Birthdate:	1936-12-17
Street:	00120 Via del Pellegrino
City:	Citta del Vaticano
State:	VC
Zipcode:	136660911
Primary Physican:	Cyrus The Great

Edit | Main Menu

Patient display output

Register as doctor

Your SSN: •••••••••

First Name: Cyrus

Last Name: The Great

Specialty: Internal Medicine

First Year in Practice: 1901

Register

Register as new doctor

Registration successful.

ID:	11
First Name:	Cyrus
Last Name:	The Great
Specialty:	Internal Medicine
First Year in Practice:	1901

Edit | Main Menu

Registered doctor output

Enter doctor id and name

ID: 11

Last Name: The Great

Get Doctor Profile

Display doctor profile

ID:	11
First Name:	Cyrus
Last Name:	The Great
Specialty:	Family Medicine
First Year in Practice:	1902

Edit | Main Menu

Doctor display output

Request a report about quantity of filled drugs.

Enter pharmacy id, start and end date for the report.

PharmacyID: 6

Start Date: 1940-01-01

End Date: 2023-05-01

Submit

Filled Drug Quantity by Pharmacy

Result:

No.	Drug	Quantity
1	Advair	150
2	Glucophage	150
3	Glucotrol	150
4	Keflex	50
5	Lopressor	200
6	Singulair	50
7	Viagra	200
8	Zetia	200

Drug Quantity by Pharmacy output

Request a report about quantity of prescribed drugs.

Enter the drug name, start and end date for the report.

There is no record.

Drug Name: lisinopril and hydrochlorothiazide

Start Date: 1930-01-01

End Date: 2023-05-01

Submit

Prescribed Drug quantity by doctor

Result:

No.	Drug	Quantity
1	Matteo Sanchez	100
2	Lydia Carter	200

Drug quantity by doctor output

Appendix II: Screenshots for Handling User Input Exceptions

Every input is checked for various validations. The following screenshots show a portion of input exception cases. More data validation information included at the bottom.

New Prescription Form

Doctor SSN should not begin with 0 or 9.

Doctor SSN: 994119411

Doctor First Name: Aryamehr

Doctor Last Name: The Shah

Patient SSN: 666666911

Patient First Name: Jimmy

Patient Last Name: Carter

Drug Name: alprazolam

Quantity: 1979

Create Prescription

Entering an invalid prescription

Message response to incorrect Doctor SSN data. Social security numbers never start with a 0 or a 9. The middle 2 digits are 01-99 (never 00). And the last 4 digits are 0001-9999 (never 0000).

Request Prescription be filled.

Enter pharmacy name, address and prescription Rx number.

Record not matched.

Rx: 103

Patient Last Name: Carter

Pharmacy Name: Jaxson Drugs

Pharmacy Address: 714 Kingston, Kayden, RI, 48863

Request Fill for Prescription

Failure when patient requests prescription be filled

Message response to incorrect Rx data.
The prescription Rx number is supposed to exist in the database.

Register as new user

The birthdate should be between 1900 to 2022.

Your SSN:

Your First Name: Jimmy

Your Last Name: Carter

Birth Date: 10/01/1899

Street: 1600 Pennsylvania Ave NW

City: Washington

State: DC

Zipcode: 20500

Primary Physician Name: Cyrus The Great

Register

Patient registration failure for invalid data

Message response to incorrect birthdate.
Year must be 4 digits in range 1900-2022.

Register as doctor

Enter 9 digits of social security number.

Your SSN: ••••

First Name: Aryamehr

Last Name: The Shah

Specialty: The King

First Year in Practice: 1919

Register

Doctor registration failure for invalid data

Message response to incorrect doctor SSN data.
Social security numbers must be 9 digits.

- Data validation checks includes:
    - Prescription quantity cannot be zero or negative. Very large numbers are also an error.
    - Names for people, cities, states cannot be a blank line or and must consist of alphabetic a-z or A-Z characters.
    - Zip Codes must be 5 or 5+4 digits.
    - Social security numbers must be 9 digits. Social security numbers never start with a 0 or a 9. The middle 2 digits are 01-99 (never 00). And the last 4 digits are 0001-9999 (never 0000).
    - Year must be 4 digits in range 1900-2022.
    - Other dates are of the format yyyy-mm-dd where mm is in the range 1-12 and dd in the range 1-31.

Conclusion

The project was a great way to experience teamwork and planning, as well as further understanding of how SQL should be formatted in both code and ER diagrams. This project has provided great experience in creating automation services for pharmacies, medical doctors and patients over prescriptions. We first had a rough draft of our ER diagram that had quite a few errors that we're glad we ran through with Professor Wisneski. Through the consultation, we've learned enhancing corrections such as the correct use of relation connections such as "one to many" and "one to one", knowing necessary connections as well as tables for the project's simplicity. We're glad to have gone through this planning correction and, with every question cleared so far, were eager to go through part 2.

Part 2 of the assignment was no walk in the park, even though we had expected it. Although using the Spring Tool Suite, which has made having a server so much easier to use, we still had a bit of an error connecting with each teammate's codes. We also had various error outputs relating to how the web browser takes in user inputs and inserts them into the database. These errors were able to be solved through group meetings by correcting details as meticulous as the input verifications. The program requires a lot of user input, and thus we are proud to have created a wide variety of input verifications. This section gave us the opportunity to manipulate HTML coding, install various input verifications, apply the MODEL-VIEW-CONTROLLER design pattern, and use a server to meet our assignment requirements. We are excited to find more challenges regarding the many applications of databases.