Otter Drug Co. Database request

Jan 24, 2022

Saul Mendoza-Loera, Yavik Kapadia, Layla Gallez

Project Overview

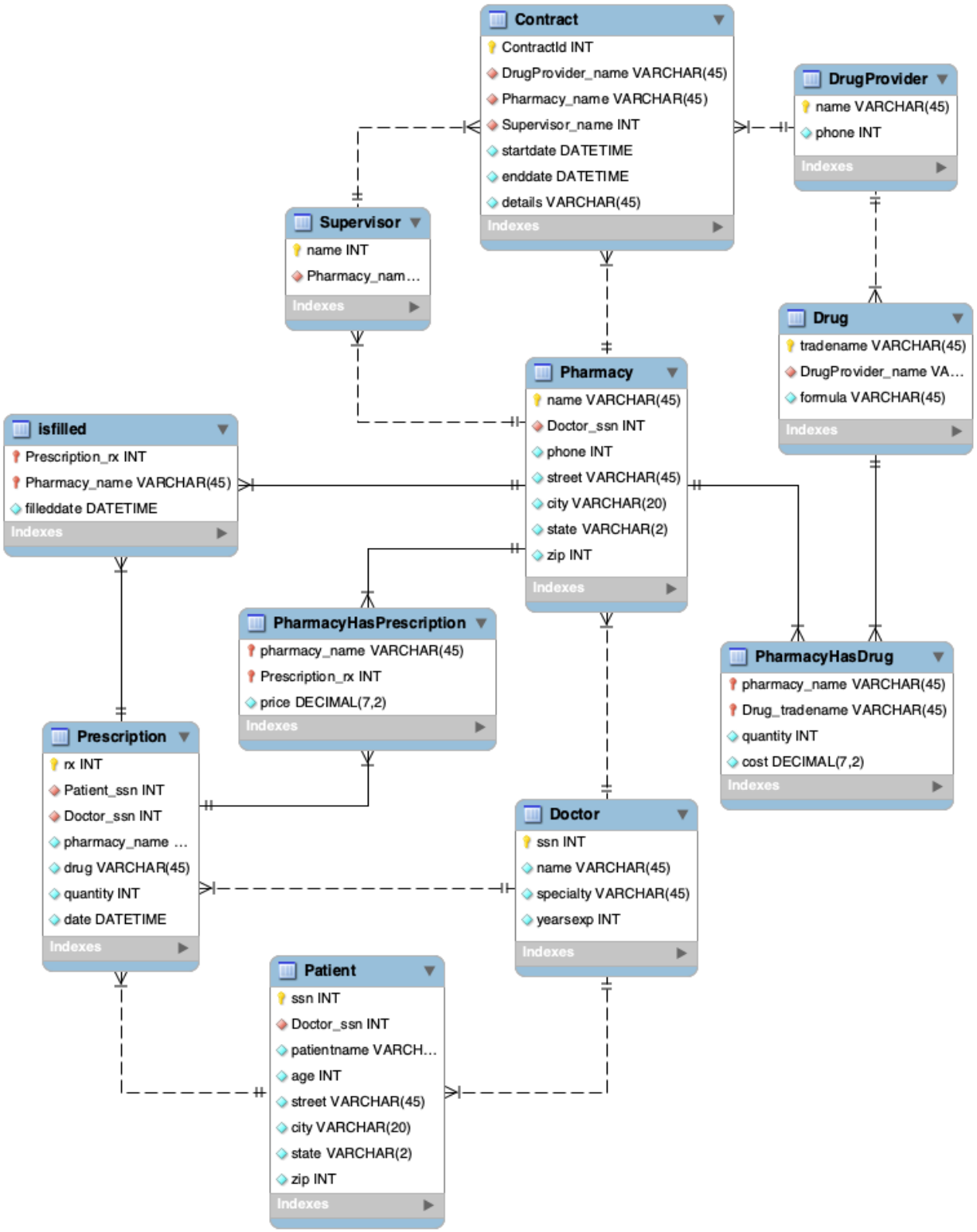
This project aims to create a relational database on behalf of Otter Drug Company, which owns the popular drug store chain OtterZone. This database will model the relationships and transactions between patients, physicians, pharmacies, and drug manufacturers. As such, Nimbus cloud services will be responsible for creating a relational database that utilizes industry-standard practices.

Based on the specifications provided by the client, the patient entity will be strongly related to the doctor entity. It would be used to store SSN, name, age, address, and the name of their primary physician. A doctor can have multiple patients and can write prescriptions for any patient. The doctor entity will contain the SSN, name, specialty, and year of experience attributes. Prescriptions created by doctors will have a unique RX number; this unique identifier will be for a specific drug and patient. The prescription entity will contain RX number, doctor name, patient name, drug name, quantity, and date prescribed, which pharmacies will use to track and fill orders.

Additionally pharmacies will have contracts with pharmaceutical companies that manufacture and supply drugs. These contracts will be overseen by a supervisor that is appointed by the pharmacy. As such, prices of drugs may vary between pharmacies. Pharmaceutical companies may have contracts with many pharmacies and pharmacies may have contracts with many pharmaceutical companies. Furthermore, a supervisor may oversee multiple contracts and a pharmacy may have multiple supervisors. The pharmacy entity will have attributes for name, phone, address, whereas the pharmaceutical company entity will have attributes for name and phone number which will be related to the drug entity. The drug entity will have attributes for generic formulas and trade name which is its unique identifier. The contract entity will record the start

data, end date, and the contract text which would be related to a singular pharmacy and a supervisor. Lastly the supervisor entity will record the name of the supervisor and pharmacy they are related to.

Model


Contract
ContractId INT
DrugProvider_name VARCHAR(45)
Pharmacy_name VARCHAR(45)
Supervisor_name INT
startdate DATETIME
enddate DATETIME
details VARCHAR(45)
Indexes
DrugProvider
name VARCHAR(45)
phone INT
Indexes
Supervisor
name INT
Pharmacy_nam...
Indexes
Drug
tradename VARCHAR(45)
DrugProvider_name VA...
formula VARCHAR(45)
Indexes
Pharmacy
name VARCHAR(45)
Doctor_ssn INT
phone INT
street VARCHAR(45)
city VARCHAR(20)
state VARCHAR(2)
zip INT
Indexes
isfilled
Prescription_rx INT
Pharmacy_name VARCHAR(45)
filleddate DATETIME
Indexes
PharmacyHasPrescription
pharmacy_name VARCHAR(45)
Prescription_rx INT
price DECIMAL(7,2)
Indexes
PharmacyHasDrug
pharmacy_name VARCHAR(45)
Drug_tradename VARCHAR(45)
quantity INT
cost DECIMAL(7,2)
Indexes
Prescription
rx INT
Patient_ssn INT
Doctor_ssn INT
pharmacy_name ...
drug VARCHAR(45)
quantity INT
date DATETIME
Indexes
Doctor
ssn INT
name VARCHAR(45)
specialty VARCHAR(45)
yearsexp INT
Indexes
Patient
ssn INT
Doctor_ssn INT
patientname VARCH...
age INT
street VARCHAR(45)
city VARCHAR(20)
state VARCHAR(2)
zip INT
Indexes

Our approach for designing the Pharmacy's database model was to apply our understanding of the client's requirements into an enhanced entity-relationship EER diagram. We developed eleven tables to identify each entity. The Patient, Doctor, Prescription, Pharmacy, Drug, DrugProvider, Contract, and Supervisor tables all have a primary key that uniquely identifies each entity. Each table has the ability to be updated with new or deleted records without compromising the data of the other entities.

The other three tables, isFilled, PharmacyHasPrescription, and PharmacyHasDrug are dependent on their related entities. Each line connecting the tables in the diagram indicates the relationships between entities. For example, the Doctor table is related to the Patient table with a crow's-feet-style relationship. This allows any physician in the Doctor's table to write many prescriptions to many patients.

We've made assumptions based on the Pharmacy's requirements. The requirement for addresses was split into separate columns for their respective tables. This decision was made to add functionality to geographical analysis. Price and Cost attributes were set to seven integers and two integers after the decimal point. This decision was made to allow for expensive drugs in high quantities. We've set all attributes of all tables to not null values. This decision was made to ensure the security of the patient, if personally identifiable information is mismatched, this could indicate errors or malicious activity.

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_SUBSTITUTION';

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

-- -----------------------------------------------------
-- Table `mydb`.`Doctor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Doctor` (
  `ssn` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `specialty` VARCHAR(45) NOT NULL,
  `yearsexp` INT NOT NULL,
  PRIMARY KEY (`ssn`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Patient` (
  `ssn` INT NOT NULL,
  `Doctor_ssn` INT NOT NULL,
  `patientname` VARCHAR(45) NOT NULL,
  `age` INT NOT NULL,
  `street` VARCHAR(45) NOT NULL,
  `city` VARCHAR(20) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
  `zip` INT NOT NULL,
  PRIMARY KEY (`ssn`),
  INDEX `fk_Patient_Doctor1_idx` (`Doctor_ssn` ASC) VISIBLE,
  CONSTRAINT `fk_Patient_Doctor1`
    FOREIGN KEY (`Doctor_ssn`)
```

```
    REFERENCES `mydb`.`Doctor` (`ssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`DrugProvider`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`DrugProvider` (
  `name` VARCHAR(45) NOT NULL,
  `phone` INT NOT NULL,
  PRIMARY KEY (`name`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Drug` (
  `tradename` VARCHAR(45) NOT NULL,
  `DrugProvider_name` VARCHAR(45) NOT NULL,
  `formula` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`tradename`),
  UNIQUE INDEX `tradename_UNIQUE` (`tradename` ASC) VISIBLE,
  INDEX `fk_Drug_DrugProvider1_idx` (`DrugProvider_name` ASC) VISIBLE,
  CONSTRAINT `fk_Drug_DrugProvider1`
    FOREIGN KEY (`DrugProvider_name`)
    REFERENCES `mydb`.`DrugProvider` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Pharmacy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Pharmacy` (
  `name` VARCHAR(45) NOT NULL,
  `Doctor_ssn` INT NOT NULL,
  `phone` INT NOT NULL,
  `street` VARCHAR(45) NOT NULL,
  `city` VARCHAR(20) NOT NULL,
  `state` VARCHAR(2) NOT NULL,
```

```
  `zip` INT NOT NULL,
  PRIMARY KEY (`name`),
  INDEX `fk_Pharmacy_Doctor1_idx` (`Doctor_ssn` ASC) VISIBLE,
  CONSTRAINT `fk_Pharmacy_Doctor1`
    FOREIGN KEY (`Doctor_ssn`)
    REFERENCES `mydb`.`Doctor` (`ssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Prescription`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Prescription` (
  `rx` INT NOT NULL,
  `Patient_ssn` INT NOT NULL,
  `Doctor_ssn` INT NOT NULL,
  `pharmacy_name` VARCHAR(45) NOT NULL,
  `drug` VARCHAR(45) NOT NULL,
  `quantity` INT NOT NULL,
  `date` DATETIME NOT NULL,
  PRIMARY KEY (`rx`),
  INDEX `fk_Patient_has_Doctor_Patient1_idx` (`Patient_ssn` ASC) VISIBLE,
  UNIQUE INDEX `rx_UNIQUE` (`rx` ASC) VISIBLE,
  INDEX `fk_Prescription_Doctor1_idx` (`Doctor_ssn` ASC) VISIBLE,
  CONSTRAINT `fk_Patient_has_Doctor_Patient1`
    FOREIGN KEY (`Patient_ssn`)
    REFERENCES `mydb`.`Patient` (`ssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prescription_Doctor1`
    FOREIGN KEY (`Doctor_ssn`)
    REFERENCES `mydb`.`Doctor` (`ssn`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`PharmacyHasPrescription`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`PharmacyHasPrescription` (
  `pharmacy_name` VARCHAR(45) NOT NULL,
```

```
  `Prescription_rx` INT NOT NULL,
  `price` DECIMAL(7,2) NOT NULL,
  PRIMARY KEY (`pharmacy_name`, `Prescription_rx`),
  INDEX `fk_pharmacy_has_prescription_pharmacy1_idx` (`pharmacy_name` ASC) VISIBLE,
  INDEX `fk_PharmacyHasPrescription_Prescription1_idx` (`Prescription_rx` ASC) VISIBLE,
  CONSTRAINT `fk_pharmacy_has_prescription_pharmacy1`
    FOREIGN KEY (`pharmacy_name`)
    REFERENCES `mydb`.`Pharmacy` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PharmacyHasPrescription_Prescription1`
    FOREIGN KEY (`Prescription_rx`)
    REFERENCES `mydb`.`Prescription` (`rx`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`PharmacyHasDrug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`PharmacyHasDrug` (
  `pharmacy_name` VARCHAR(45) NOT NULL,
  `Drug_tradename` VARCHAR(45) NOT NULL,
  `quantity` INT NOT NULL,
  `cost` DECIMAL(7,2) NOT NULL,
  PRIMARY KEY (`pharmacy_name`, `Drug_tradename`),
  INDEX `fk_pharmacy_has_Drug_Drug1_idx` (`Drug_tradename` ASC) VISIBLE,
  INDEX `fk_pharmacy_has_Drug_pharmacy1_idx` (`pharmacy_name` ASC) VISIBLE,
  CONSTRAINT `fk_pharmacy_has_Drug_pharmacy1`
    FOREIGN KEY (`pharmacy_name`)
    REFERENCES `mydb`.`Pharmacy` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_pharmacy_has_Drug_Drug1`
    FOREIGN KEY (`Drug_tradename`)
    REFERENCES `mydb`.`Drug` (`tradename`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Supervisor`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Supervisor` (
  `name` INT NOT NULL,
  `Pharmacy_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`name`),
  INDEX `fk_Supervisor_Pharmacy1_idx` (`Pharmacy_name` ASC) VISIBLE,
  CONSTRAINT `fk_Supervisor_Pharmacy1`
    FOREIGN KEY (`Pharmacy_name`)
    REFERENCES `mydb`.`Pharmacy` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Contract`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Contract` (
  `ContractId` INT NOT NULL AUTO_INCREMENT,
  `DrugProvider_name` VARCHAR(45) NOT NULL,
  `Pharmacy_name` VARCHAR(45) NOT NULL,
  `Supervisor_name` INT NOT NULL,
  `startdate` DATETIME NOT NULL,
  `enddate` DATETIME NOT NULL,
  `details` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ContractId`),
  INDEX `fk_DrugProvider_has_Pharmacy_Pharmacy1_idx` (`Pharmacy_name` ASC) VISIBLE,
  INDEX `fk_DrugProvider_has_Pharmacy_DrugProvider1_idx` (`DrugProvider_name` ASC)
VISIBLE,
  INDEX `fk_Contract_Supervisor1_idx` (`Supervisor_name` ASC) VISIBLE,
  CONSTRAINT `fk_DrugProvider_has_Pharmacy_DrugProvider1`
    FOREIGN KEY (`DrugProvider_name`)
    REFERENCES `mydb`.`DrugProvider` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_DrugProvider_has_Pharmacy_Pharmacy1`
    FOREIGN KEY (`Pharmacy_name`)
    REFERENCES `mydb`.`Pharmacy` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Contract_Supervisor1`
    FOREIGN KEY (`Supervisor_name`)
    REFERENCES `mydb`.`Supervisor` (`name`)
    ON DELETE NO ACTION
```

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


-- -----------------------------------------------------
-- Table `mydb`.`isfilled`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`isfilled` (
  `Prescription_rx` INT NOT NULL,
  `Pharmacy_name` VARCHAR(45) NOT NULL,
  `filleddate` DATETIME NOT NULL,
  PRIMARY KEY (`Prescription_rx`, `Pharmacy_name`),
  INDEX `fk_isfilled_Pharmacy1_idx` (`Pharmacy_name` ASC) VISIBLE,
  INDEX `fk_isfilled_Prescription1_idx` (`Prescription_rx` ASC) VISIBLE,
  CONSTRAINT `fk_isfilled_Pharmacy1`
    FOREIGN KEY (`Pharmacy_name`)
    REFERENCES `mydb`.`Pharmacy` (`name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_isfilled_Prescription1`
    FOREIGN KEY (`Prescription_rx`)
    REFERENCES `mydb`.`Prescription` (`rx`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


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

Test Queries

-- 1 Display the drug that costs less than the average of all drugs sold by all pharma companies.
-- Sort by drug.

-- Explanation: This question is asking for one column (Drug_tradename) from the table
-- PharmacyHasDrug since the table holds the cost of each drug. A subquery must take place to determine what the average cost is from all
-- the drugs being sold.

```
SELECT DISTINCT Drug_tradename
FROM PharmacyHasDrug
WHERE cost < (SELECT AVG(Drug_tradename) FROM PharmacyHasDrug)
ORDER BY Drug_tradename;
```

-- 2 Display the pharmacies that sell the following drugs: Neurotin, Altabax, and Lenvina.
-- Display the doctor's specialty who has prescribed these drugs.

-- Explanation: This question is asking for two columns (pharmacy_name and specialty) from two tables (prescription and doctor).
-- Using JOIN and IN will display the final answer.

```
SELECT prescription.pharmacy_name, doctor.specialty
FROM prescription join doctor on prescription.Doctor_ssn = Doctor.ssn
WHERE prescription.drug IN ('Neurotin', 'Altabax', 'Lenvina');
```

-- 3 Display the name of the drug provider who does not have their drugs sold in the OtterZone pharmacy.
-- SORT BY name of drug provider

-- Explanation: This question is asking for one column (name) of the drug provider. You must use DrugProvider, Drug, and PharmacyHasDrug
-- in multiple JOINS to determine what drug is not sold in the OtterZone pharmacy. != operation will be used to prevent the drug that is sold at OtterZone.

```
SELECT DrugProvider.name
FROM DrugProvider join Drug on DrugProvider.name = Drug.DrugProvider_name
        join PharmacyHasDrug on PharmacyHasDrug.tradename =
PharmacyHasDrug.DrugProvider_name
WHERE PharmacyHasDrug.pharmacy_name != 'OtterZone'
ORDER BY DrugProvider.name;
```

-- 4 Display doctor's SSN and name who specialize in Neurology or Cardiology.
-- Doctors should have atleast 5 years of experience in their field.

-- Explanation: This question is asking for the ssn and name of the doctor. UNION will happen to grab data from the doctor table that specialize in
-- Neurology or Cardiology. AND operation will also be used to take into account that the doctor will need at least 5 years of experience.

```
SELECT ssn, name
FROM doctor
WHERE specialty = 'Neurology' AND yearsexp >= 5
UNION
SELECT ssn, name
FROM doctor
WHERE specialty = 'Cardiology' AND yearsexp >= 5;
```

-- 5 Display number of times OtterZone has filled the prescription for the drug Cottelic.

-- Explanation: This question is asking for the count of Prescription_rx. Since Prescription_rx is unique to the prescription, you
-- can use subquery to obtain the data of all rx that are for the drug 'Cottelic'.

```
SELECT COUNT(Prescription_rx)
FROM isFilled
WHERE Pharmacy_name = 'OtterZone' AND Prescription_rx IN (SELECT rx FROM
Prescription WHERE drug = 'Cottelic')
GROUP BY prescription.drug;
```

Conclusion

We've developed a diagram that represents the requirements provided by the Pharmacy. We then converted the diagram into a database schema that creates the database, tables, and their relationships. These tables and relationships were then analyzed and used to create meaningful test queries that show the potential uses of the database.

This project gave us the opportunity to get familiar with database diagrams and schemas. Being mindful of how each entity should interact with one another is important for functionality. Open questions that still need to be answered in part two include the possible need for views, will our test queries require larger attribute length, and will our common column names cause confusion with joins.