CST 363: Intro to Database Systems - Project 2 Database Design

Drug Store Chain Database & Java Web Application

Team AlphaGamez

May 2021

Created By:

Fadl Ghaddar

Abraham Borg

Benjamin Mona

Introduction

The database outlined in this report is meant to store information for an expanding drugstore company. The database is designed to manage information about patients, doctors, and prescriptions for medication. The database will be flexible enough to accommodate many different pharmacies and be able to record the drugs supplied by that pharmacy as well as the prices of each drug for a given pharmacy. The drug store chain will also store the information about contracts between different pharmacies and different pharmaceutical companies. Each contract will be managed by a contract supervisor.

The drugstore company needs the database to successfully manage the following process. Each patient is assigned a primary care physician (PCP). All PCP's are part of a list of in-network doctors. All patients can have many doctors, and all doctors may have many patients. Each doctor may prescribe medication for a patient. This prescription is stored in the database and a pharmacy is assigned to fill that prescription. All relevant information about prescriptions and fill dates are saved each time a prescription is made. The pharmacy assignment to fill the prescription is determined by which pharmacy is stocked with the prescribed medication, and also by the best price. Therefore, a catalog of pharmaceutical companies and their pharmaceutical products are recorded in the database. Finally, each pharmacy may have one or more contracts with pharmaceutical companies and vice-versa. The database will accommodate for any changes that occur when a pharmaceutical company is no longer a partner of the drugstore chain. This means that all medications that are supplied by the pharmaceutical company will be removed from the database.

Finally, a web application has been implemented for the client with four operations that are critical to the client's business plan. The four operations available in the web application are:
1) write a new prescription, 2) request a prescription to be filled, 3) generate a report on drug usage, and 4) generate a report for the FDA.

ER Diagram

The implementation of this database required careful application of relationships between entities and precise constraints to ensure that several key operations on the database produced correct results. The ER diagram describes the overall structure and relationships between entities. The diagram is color-coded to indicate the different attributes and the primary keys are also shown. Many things which could not be shown on the ER diagram are described below in further detail.

The Foreign Key “Pharmaceutical_Company_Name” from the table “drug” references “Name” from “Pharmaceutical_Company”, and this relationship operates on “on delete cascade” and “on update cascade”. This ensures that drugs are removed or updated when a pharmaceutical company is removed or updated within the database. The attribute “PCP_SSN” is a Foreign Key that references the “SSN” of the doctor within the doctor table. The Foreign Key “PCP_SS” has the following constraints: “on delete restrict” and “on update cascade”. These constraints ensure that a doctor cannot be removed from the database if the doctor is a PCP to a patient until the patient has been assigned another PCP. Foreign Key “Pharmaceutical_Company_Name” from the table “pharmacy_pharmaceutical_contract” references “Name” from “Pharmaceutical_Company”, and Foreign Key “Pharmacy_Name” from “pharmacy_pharmaceutical_contract” references “Name” from the table “Pharmacy”. The restrictions applied to both of these references are “on delete cascade” and “on update cascade”. These constraints ensure that if a pharmaceutical company or a pharmacy is deleted, then the contract between the two will also be deleted. Also, if a pharmaceutical company or a pharmacy changes its names, then contracts signed by either will also reflect those changes. Foreign Key “Drug_name” from “pharmacy_has_drug” references “Name” from the “drug” table, and foreign key “Pharmacy_Name” from “pharmacy_has_drug” references “Name” from the “pharmacy” table. Both these references operate by “on delete cascade” and “on update cascade”.

ER Diagram (continued)

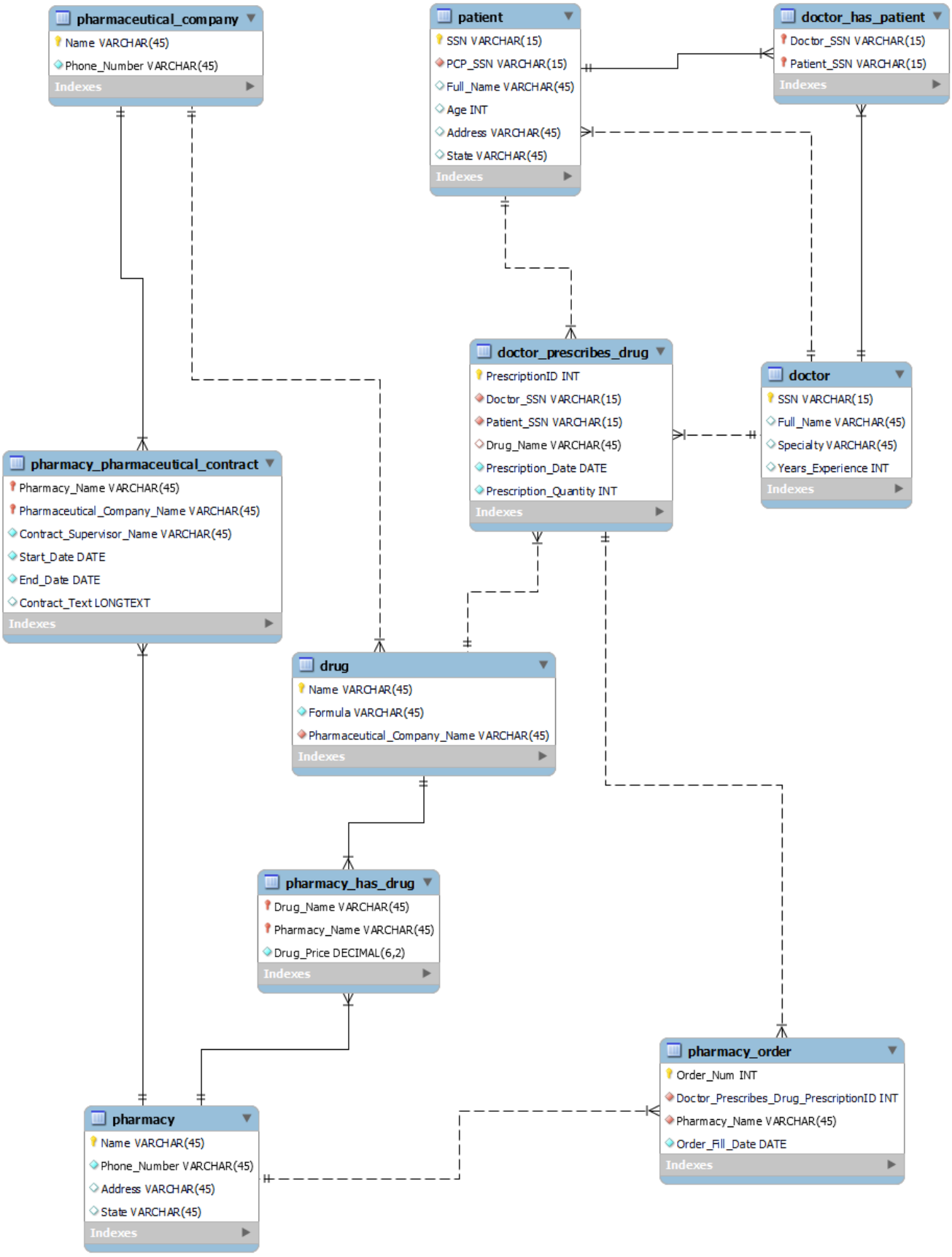
These constraints ensure that if a drug or a pharmacy is deleted/updated, then those records will be deleted or updated in the will be reflected in the “Pharmacy_has_drugs” table. If a drug is deleted from within the “drug” table, then the “doctor_prescribes_drug” table will set that “drug_name” to null in its record. This happens because the relationship between the two operates on “on delete set null.” Attempts to delete a patients SSN or a doctors SSN will be restricted by the “doctor_prescribes_drug” table. If a patient SSN or doctor SSN is updated, those changes will be reflected in the “doctor_prescribes_drug” table. If a “PrescriptionID” from “doctor_prescribes_drug” is deleted or updated, then those changes will cascade onto the “pharmacy_order” table.

The Foreign Keys “Doctor_SSN” and “Patient_SSN” in “Doctor_has_patient” references “SSN” within the “patient” table and “SSN” within the doctor table. Attempts to delete “SSN” within the patient or doctor table will be restricted because the relationship between the keys operate on a “on delete restrict” basis. However, updating an SSN in either the “doctor” or the “patient” table will result in updates cascading down to the “Doctor_SSN” and “Patient_SSN” foreign keys.

Finally, we will need to write application code that ensures that the all SSN are in the format “xxx-xxx-xxx.” Likewise, all phone numbers must be within the format “xxx-xxx-xxxx”, and the application code must ensures it happens. Lastly, the application code should also ensure that if a doctor prescribes the same drug for the same patient, the previous prescription will be deleted from the database, and the latest prescription be be stored in its place.

It was due to these careful and thorough applications of constraints and the correct cardinality between the various entities that the database achieves the goals of the drug store client.

ER Diagram (continued)



Relational Schema

```
DROP SCHEMA IF EXISTS Project2_AlphaGamez;
CREATE SCHEMA Project2_Alphagamez;
DROP DATABASE IF EXISTS Project2_AlphaGamez;
CREATE DATABASE Project2_AlphaGamez;
USE Project2_AlphaGamez;

CREATE TABLE doctor
(
  `SSN` VARCHAR(15) NOT NULL,
  `Full_Name` VARCHAR(45) NULL,
  `Specialty` VARCHAR(45) NULL,
  `Years_Experience` INT NULL,
  PRIMARY KEY (SSN)
);

CREATE TABLE patient
(  `SSN` VARCHAR(15) NOT NULL,
  `PCP_SSN` VARCHAR(15) NOT NULL,
  `Full_Name` VARCHAR(45) NULL,
  `Age` INT NULL,
  `Address` VARCHAR(45) NULL,
  `State` VARCHAR(45) NULL,
   PRIMARY KEY (SSN),
      FOREIGN KEY (PCP_SSN) REFERENCES doctor(SSN) ON DELETE RESTRICT ON UPDATE CASCADE
);

CREATE TABLE Pharmaceutical_Company
(
`Name` VARCHAR(45) NOT NULL,
`Phone_Number` VARCHAR(45) NOT NULL,
 PRIMARY KEY (Name)
);

CREATE TABLE Drug
(
    `Name` VARCHAR(45) NOT NULL,
    `Formula` VARCHAR(45) NOT NULL,
    `Pharmaceutical_Company_Name` VARCHAR(45) NOT NULL,
    PRIMARY KEY (Name),
FOREIGN KEY (Pharmaceutical_Company_Name) REFERENCES Pharmaceutical_Company (Name) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Pharmacy
(
  `Name` VARCHAR(45) NOT NULL,
  `Phone_Number` VARCHAR(45) NOT NULL,
  `Address` VARCHAR(45) NULL,
  `State` VARCHAR(45) NULL,
  PRIMARY KEY (Name)
);

CREATE TABLE Pharmacy_has_Drug
(
`Drug_Name` VARCHAR(45) NOT NULL,
`Pharmacy_Name` VARCHAR(45) NOT NULL,
`Drug_Price` DECIMAL(6,2) NOT NULL,
PRIMARY KEY (Drug_Name, Pharmacy_Name),
FOREIGN KEY (Pharmacy_Name) REFERENCES Pharmacy(Name) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (Drug_Name) REFERENCES Drug(Name) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Relational Schema (Continued)

```
CREATE TABLE Doctor_Prescribes_Drug
(
  `PrescriptionID` INT NOT NULL AUTO_INCREMENT,
  `Doctor_SSN` VARCHAR(15) NOT NULL,
  `Patient_SSN` VARCHAR(15) NOT NULL,
  `Drug_Name` VARCHAR(45) NULL,
  `Prescription_Date` DATE NOT NULL,
  `Prescription_Quantity` INT NOT NULL,
   PRIMARY KEY (PrescriptionID),
      FOREIGN KEY (Doctor_SSN) REFERENCES doctor(SSN) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Patient_SSN) REFERENCES patient(SSN) ON DELETE RESTRICT ON UPDATE CASCADE,
FOREIGN KEY (Drug_Name) REFERENCES Drug(Name) ON DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE Pharmacy_Order
(
`Order_Num` INT UNSIGNED NOT NULL AUTO_INCREMENT,
`Doctor_Prescribes_Drug_PrescriptionID` INT NOT NULL,
`Pharmacy_Name` VARCHAR(45) NOT NULL,
`Order_Fill_Date` DATE NOT NULL,
PRIMARY KEY (Order_Num),
   FOREIGN KEY (Pharmacy_Name) REFERENCES Pharmacy(Name) ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (Doctor_Prescribes_Drug_PrescriptionID) REFERENCES Doctor_Prescribes_Drug(PrescriptionID)
   ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Pharmacy_Pharmaceutical_Contract
(
 `Pharmacy_Name` VARCHAR(45) NOT NULL,
 `Pharmaceutical_Company_Name` VARCHAR(45) NOT NULL,
 `Contract_Supervisor_Name` VARCHAR(45) NOT NULL,
 `Start_Date` DATE NOT NULL,
 `End_Date` DATE NOT NULL,
 `Contract_Text` LONGTEXT NULL,
  PRIMARY KEY (Pharmacy_Name, Pharmaceutical_Company_Name),
  FOREIGN KEY (Pharmacy_Name) REFERENCES Pharmacy(Name) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (Pharmaceutical_Company_Name) REFERENCES Pharmaceutical_Company(Name)
  ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Doctor_has_Patient
(
  `Doctor_SSN` VARCHAR(15) NOT NULL,
  `Patient_SSN` VARCHAR(15) NOT NULL,
   PRIMARY KEY (Doctor_SSN, Patient_SSN),
   FOREIGN KEY (Doctor_SSN) REFERENCES doctor(SSN) ON DELETE RESTRICT ON UPDATE CASCADE,
   FOREIGN KEY (Patient_SSN) REFERENCES patient(SSN) ON DELETE RESTRICT ON UPDATE CASCADE
);
```

Normalized Relational Schema

The initial ER Diagram was well constructed but there were two issues. The first came from table doctor_prescribes_drug, and the second issue was from the table drug. Our first run on the doctor_prescribes_drug was incorrect as it was in 1NF from. The primary key for doctor_prescribes_drug was a composite key made up of Doctor_SSN and Patient_SSN.
This composite key was then used to retrieve the Drug_Name, Prescription_Date, and Prescription_Quantity values. The problem with this approach was that the database assumed that a specific doctor can only prescribe one Drug_Name to their specific patient at one time, when that is not the case. To counter this problem, we changed the structure of the doctor_prescribes_drug table by altering its primary key. We ensured that the primary key for doctor_prescribes_drug was just one key named PrescriptionID, and this unique key then indicated the following information: SSN of the doctor that prescribed it (Foreign Key), SSN of the patient it was prescribed to (Foreign Key), the Drug_Name (Foreign Key). Prescription_Date, and Prescription_Quantity. Having a unique identifier for each prescription ensured that doctors are able to issue different and multiple drugs to each of their patients.

Changes we made from the doctor_prescribes_drug table cascaded into changes that we made in the pharmacy_order. Since our initial version of pharmacy_order table referenced Doctor_SSN, Patient_SSN, and Drug_Name from the doctor_prescribes_drug table, all we had to do was change pharmacy_order so that it referenced PrescriptionID instead of those 3 keys.
Finally, the last change we made within our ER Diagram came from the drug table.
Initially, this table had two composite keys that made up the primary key, and these two composite keys were Drug_Name and Formula. We made changes to the table that resulted in Drug_Name becoming the primary key.

Java Web Application

A Java web application has been implemented for the client using Spring Boot. The client outlined four critical applications that needed to be available in the web application, see Figure 1. These four operations are described below.

The first operation available in the web application assists doctors to write prescriptions for their patients. This form is shown in Figure 2. The output of this operation will be the prescription identification number and the prescription data, see Figure 3. The information shown in the figure is valid and therefore it will be added to the database, see Figure 4. This is a secure system so the doctor must input their personal information as well as the patient's identifying information. The web form rejects any request that contains information that does not match current database information, see Figure 5.

The second operation available in the web application assists the patients to make a request to a pharmacy to fill their prescription. The patient must enter valid information otherwise the prescription request will be rejected. See Figure 6 for the form and required inputs and see Figure 7 for the results of a successful operation.

The third operation available assists a pharmacy manager to request a report of the quantity of drugs that have been used to fill prescriptions by the pharmacy. See Figure 8 for the form and required inputs and Figure 9 for the resulting output. Once again, if the form receives incorrect invalid inputs, the operation will fail, and no report will be generated.

The final operation available in the web application is for an FDA official who wants to know the quantity of drugs that each doctor has prescribed. The drug name and date range must be valid. The web form and the output is shown in Figure 10 and Figure 11.

Java Web Application (Continued)

The Java web application utilized Spring Boot and a JDBC connection. PreparedStatements were used for a specific queries, and after that PreparedStatements inserted values into the database using .setString, .setInt, and .setDouble. After that, we executed the query using .executeQuery, and depending on the needs of the problem, we saved the results of that query onto a ResultSet Object.

For example, If we were receiving information from the database, such as all the drug names, then we would scan through the ResultSet Object to find if the users drug name matches with a drug name in the database. In addition, .executeUpdate was also used to insert data into the database.

Finally, we closed the connection, added the relevant data and strings to the model using the .addAttribute method, and returned a String that signified a specific HTML page

Java Web Application Figures

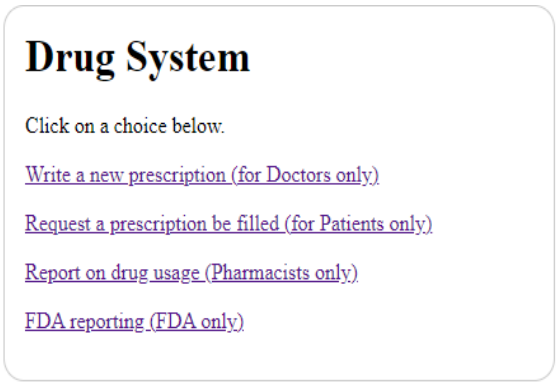


Figure 1: Web Application Home Panel

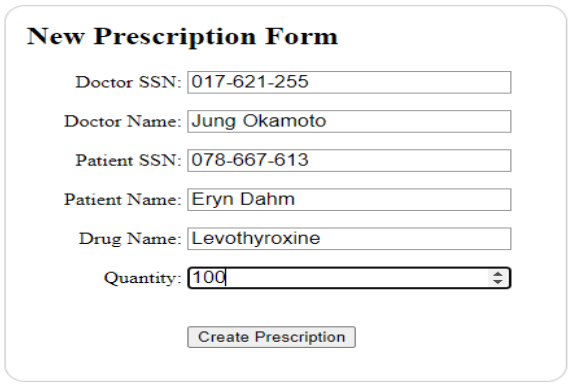


Figure 2: New Prescription Form with valid input.

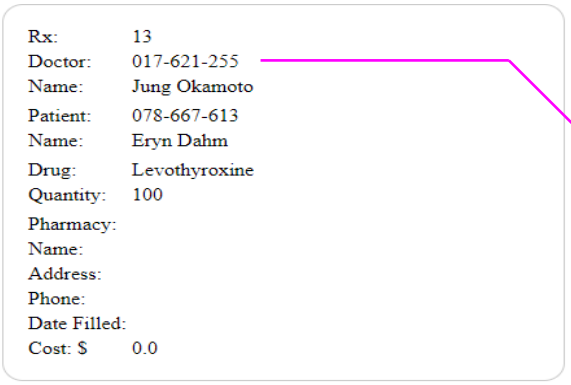
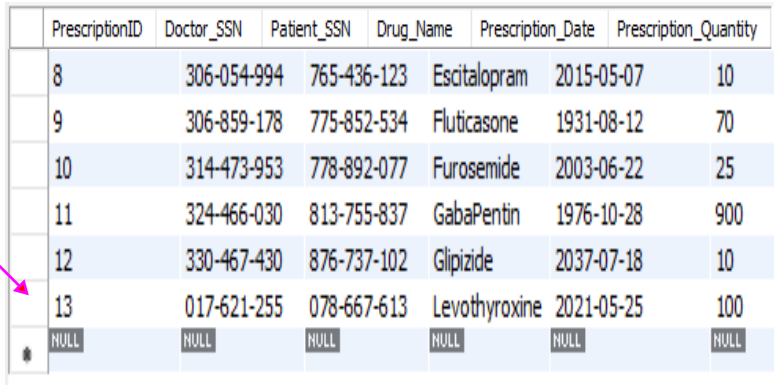


Figure 3: New Prescription Form output..



PrescriptionID	Doctor_SSN	Patient_SSN	Drug_Name	Prescription_Date	Prescription_Quantity
8	306-054-994	765-436-123	Escitalopram	2015-05-07	10
9	306-859-178	775-852-534	Fluticasone	1931-08-12	70
10	314-473-953	778-892-077	Furosemide	2003-06-22	25
11	324-466-030	813-755-837	GabaPentin	1976-10-28	900
12	330-467-430	876-737-102	Glipizide	2037-07-18	10
13	017-621-255	078-667-613	Levothyroxine	2021-05-25	100
NULL	NULL	NULL	NULL	NULL	NULL

Figure 4: New prescription successfully added to database.

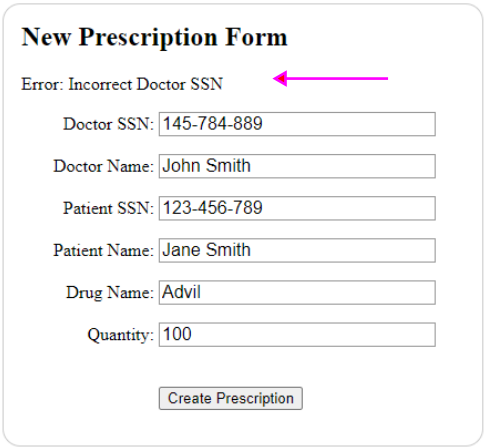


Figure 5: Prescription Form, result of invalid input.

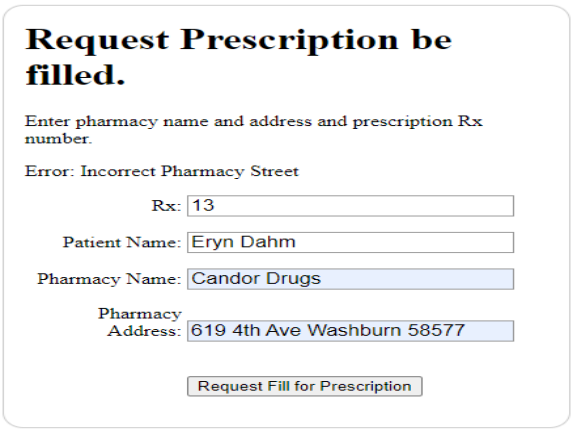


Figure 6: Request Prescription be filled form

Java Web Application Figures (Continued)

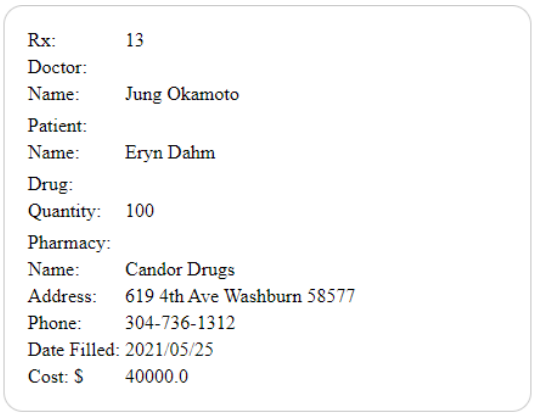


Figure 7: Prescription request form output

Figure 8: Pharmacy Drug Report Form

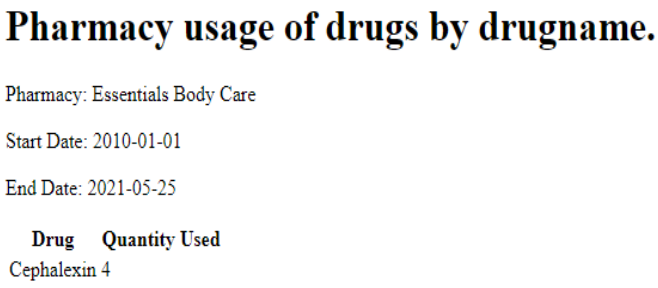


Figure 9: Pharmacy Drug Report Output

Figure 10: FDA Report Form

Figure 11: FDA Report

SQL Queries

To put the database to the test we came up with five statements and created queries to provide the correct results.

1. Select the doctor who has the highest amount of experience in “Nephrology”

Query -

```
SELECT  SSN, Full_Name, Specialty, MAX(Years_Experience)
AS "Years of Experience"
FROM Doctor WHERE Specialty = "Nephrology";
```

2. Show all prescriptions and prescription quantities as well as the name of the medication, for all patients and include the doctors who prescribed it. Order the data by date of prescription.

Query -

```
SELECT dpd.Prescription_Date, dpd.Prescription_Quantity, dpd.Drug_Name, pat.Full_Name
AS 'Patient FullName', d.Full_Name AS 'Doctor Full_Name'
FROM doctor d
JOIN doctor_has_patient dhp ON dhp.Doctor_SSN = d.SSN
JOIN doctor_prescribes_drug dpd ON dpd.Doctor_SSN = dhp.Doctor_SSN
JOIN patient pat ON pat.SSN = dhp.Patient_SSN
ORDER BY Prescription_Date;
```

3. Show medication names and price, where the contract supervisor is Charlotte Burns. Show the name of the pharmaceutical company as well as the name of the pharmacy that has the drug. No duplicate entries. Order the data by medication price.

Query -

```
SELECT DISTINCT d.name, phd.drug_price, phd.Pharmacy_Name AS 'Pharmacy Name',
ppc.Pharmaceutical_Company_Name, ppc.Contract_Supervisor_Name
FROM drug d
JOIN pharmacy_has_drug phd ON phd.Drug_Name = d.name
JOIN pharmacy p ON p.name = phd.Pharmacy_Name
JOIN pharmacy_pharmaceutical_contract ppc ON ppc.Pharmacy_Name = p.Name
WHERE ppc.Contract_Supervisor_Name = 'Charlotte Burns'
ORDER BY phd.drug_price;
```

SQL Queries (continued)

4. Show the full patient name, social security number, age, as well as medication name and formula. make sure the formula does not contain the ingredient C1. Order the data by patient age.

Query -

```
SELECT d.name AS 'Medication Name', d.formula AS 'Medication Formula', pat.SSN
AS 'Patient SSN', pat.Full_Name AS 'Patient Full Name',
pat.Age AS 'Patient Age'
FROM drug d
JOIN doctor_prescribes_drug dpd ON dpd.Drug_Name = d.Name
JOIN patient pat ON pat.SSN = dpd.Patient_SSN
WHERE d.formula NOT LIKE '%C1%'
ORDER BY pat.Age
```

5. Show Pharmacy name, city, and state as well as matching patient data. This query is used to provide patients with information on which pharmacy to order from.

Query -

```
SELECT p.name AS 'Pharmacy Name', p.Address AS 'Pharmacy Location', p.State AS
'Pharmacy State Location', pat.Full_Name AS 'Patient Full Name',
pat.State AS 'Patient Resides In St'
FROM pharmacy p
JOIN pharmacy_has_drug phd ON phd.Pharmacy_Name = p.Name
JOIN drug d ON d.Name = phd.Drug_Name
JOIN doctor_prescribes_drug dpd ON dpd.Drug_Name = d.Name
JOIN patient pat ON pat.SSN = dpd.Patient_SSN
ORDER BY p.State
```

Conclusions

As a professional consulting team we were able to create a MySQL database that met all the requirements. We created a ER Diagram for our database which we then checked for inconsistencies and normalization issues. We feel that after thorough investigation and testing we are confident that our schema met the needs of our customer. We then created test queries and challenged ourselves to find errors with our database. The Java web application was also tested throughly and will meet the client's application needs.

We feel that there are no questions that need to be answered or unsolved dilemmas remaining. As a team we have learned about utilizing MySQL and working with diagrams as well as the built in features of MySQL. This will surely assist us in our future projects and allow us to be prepared to tackle even more complex problems.