

MobiusRetrieval

PROJECT REPORT

CST 363-30: Introduction to Database

Spring 2023

Severin Light, Martin Sanchez Sendiz, Nathan Simpson

CST 363: Databases – Project Database Design and Implementation

Project Overview

Here at MobiusRetrieval we care about the ease of use and access of data. This database is built for a hypothetical drugstore chain and therefore should be useful and applicable to other drugstores and pharmacies with a similar data recording structure. With the implementation of a database, data can easily be created and managed. This database is designed to meet following customer requirements:

- Every patient will be required to provide their social security number (SSN), full name, age, and their permanent address.
- Doctors will also be required to provide their SSN, full name, their specialty, and their years of experience.
- Pharmaceutical companies will be identified by their names and each one has a phone number
- All prescription drugs are manufactured and distributed by pharmaceutical companies to retail pharmacies. Each drug has a “generic” name and can possibly have a trade name.
- Every pharmacy must have a name, address, and a telephone number.
- Each patient must have a primary care physician.
- All physicians are able to write prescriptions for any patient.
- A drug can be sold at multiple pharmacies with prices varying from pharmacy to pharmacy. Each pharmacy can sell several drugs.
- Doctors can prescribe drugs to patients and a patient is allowed to have several prescriptions from more than one doctor. Every prescription contains a unique RX number that is written by a single doctor, for one drug, and is for a single patient only.
- Each prescription comes with a date and a quantity. The prescription is for a specific drug from a specific pharmaceutical company if the prescription bears a trade name. Any drug with that formula name from any pharmaceutical company may be used if the prescription is for a generic name.
- When a prescription is filled, we will keep track of the pharmacy that filled it, the date it was filled, and, if the medication has a generic name, then the name of the pharmaceutical company that provided the medication.

CST 363: Databases – Project Database Design and Implementation

- Pharmacies and pharmaceutical companies are contracted with each other. A pharmaceutical company may have many contracts with different pharmacies as well as a pharmacy may have multiple contracts with numerous pharmaceutical companies. For every contract, we will record every start date, end date, and the terms of the contract.
- For every contract, a pharmacy will assign a supervisor. Each contract supervisor may change with time. A supervisor can supervise multiple contracts at the same time.

Database Structure

The database schema for this project consists of 11 tables. The main primary tables in this database are the patient, doctor, specialty, prescription_drug, prescription_order, pharmaceutical_company, retail_pharmacy, and supervisor tables. All other tables reference the primary tables through foreign keys. Appendix A provides an entity-relationship diagram (ERD) for this database.

Database Table	Description
Patient	Holds all of the individual patients information
Doctor	Holds all of the individual doctors information
Specialty	Contains data on doctor specialties
Prescription_Order	Holds all of the information pertaining to specific prescriptions ordered by doctors to specific patients
Retail_Pharmacy	Holds the general information of the retail pharmacy
Supervisor	Contains the data associated with the contract supervisor
Prescription_Drug	Holds the general information of the prescribed drug
Prescription_Drug_has_Retail_Pharmacy	Holds the information to determine if the prescribed drug has a retail pharmacy seller
Pharmaceutical_Company_has_Retail_Pharmacy1	Holds detailed information pertaining to the pharmaceutical company's relationship with retail pharmacies
Pharmaceutical_Company_has_Prescription_Drug	Holds the information to determine if the pharmaceutical company has the prescribed drug

CST 363: Databases – Project Database Design and Implementation

Pharmaceutical_Company	Holds the general information of the pharmaceutical company
------------------------	---

ER Model

The ERD in Appendix A illustrates how we plan to implement the customer requirements listed above. This ERD utilizes crow's foot notation to describe entity relations. Below, we will explain how each table was designed and how its relations were determined from the list of customer requirements.

Patient Table

Based on the customer requirements, the necessary patient information that needed to be recorded were the patient's SSN, name, age, and address. Additionally each patient should have a primary physician assigned to them. The table below provides a list of the fields contained in the patient table.

Name	Data Type	Constraints	Description
SSN	INT	Primary Key, Not Null, Unique	Patient's social security number used as the primary key
firstName	VARCHAR(30)	Not Null	Patient's first name
lastName	VARCHAR(30)	Not Null	Patient's last name
dateOfBirth	DATETIME	NotNull	Patient's date of birth
address	VARCHAR(256)	NotNull	Patient's permanent address
primary_physician	INT	Foreign Key, Not Null	Patient's primary physician

From the list of requirements, the patient has an identifying SSN, a name, age, and address. The patient's SSN was chosen as the primary key for the table since it's a unique identification number. Although the data type for SSN is listed as INT, the associated database application will need to contain code to restrict SSN values such that they contain 9 digits. In order to meet the patient name requirement, we decided to split up the name field into 2 separate columns, firstName and lastName. The data type of VARCHAR(30) was chosen for the name fields as we expect most names to be less than or equal to 30 characters. Additionally, while the firstName and lastName fields utilize the NotNull constraint. These name columns and assumptions are also implemented in the doctor table to keep our database design consistent.

CST 363: Databases – Project Database Design and Implementation

In order to meet the patient age requirement, we decided to record the patient's date of birth in the patient table rather than recording a value for the age. This will allow us to calculate the patients age and will remove the need to continually update the patients' ages.

The address field was set to have a data type of VARCHAR(256) to allow for an alphanumeric string to be entered for the patient's address. This parameter will hold the patient's street address, city, state, zip. The database application will need to contain code to require each of these items and concatenate them before inserting or updating addresses to the database.

Lastly, since each patient has a primary physician, the database ERD represents this using the symbol for a 1:n relationship between the doctor and patients table. That is, although each patient has one primary physician, each doctor can have multiple patients that list them as their primary physician. Within the patient's table itself, the primary_physician column is used as a primary key containing an integer value for the doctor's SSN in order to reference the appropriate entry in the doctor table.

Doctor Table

The doctor table follows a similar format to that used for the patient table. The doctor's SSN is used as a primary key since it's a unique identifying value. Like the patient table, the doctor's name requirement is fulfilled by two attributes: firstName and lastName. In order to maintain consistency throughout our database, the same restrictions are carried over from the patient table.

Name	Data Type	Constraints	Description
SSN	INT	Primary Key, Not Null, Unique	Doctor's social security number used as the primary key
firstName	VARCHAR(30)	Not Null	Doctor's first name
lastName	VARCHAR(30)	Not Null	Doctor's last name
specialty	INT	Foreign Key, Not Null	Doctor's specialty
startDate	DATETIME	NotNull	Doctor's career start date

Instead of recording the doctor's years of experience, the database records the doctor's career start date (i.e., the date they became licensed and began practicing medicine). This will allow us to calculate the years of experience instead of continually updating the value for all doctors in the table. Lastly, the doctor's specialty is listed as an INT which references the specialty table. This was done to meet 3NF since multiple doctors can share the same specialty.

Specialty Table

CST 363: Databases – Project Database Design and Implementation

The specialty table is very simple and contains two columns: ID and name. The ID column is of INT value and is used as the primary key for the table. The name column contains the specialty name as a VARCHAR(45). If the customer wishes to add additional information relating to the speciality in the future, this database structure will allow them to record the information in one location rather than having to add it in every row containing said specialty within the doctor's table.

As mentioned above, the relationship between the specialty and doctor tables is displayed as a 1:many relationship. This is done so because many doctors can share the same specialty.

Retail_Pharmacy Table

In order to meet the customer requirements for retail pharmacy information, the retail_pharmacy table contains the following columns and data constraints.

Name	Data Type	Constraints	Description
retailPharmacyID	INT	Primary Key, Not Null, Unique	Primary key for retail pharmacies
name	VARCHAR(45)	Not Null	Name of pharmacy
address	VARCHAR(256)	Not Null	Pharmacy location
phoneNumber	VARCHAR(20)	Not Null	Pharmacy phone number

An integer value representing the retail pharmacy ID was chosen as the primary key for the retail_pharmacy table. The data type of VARCHAR(45) was chosen to store the name of the pharmacy. Additionally, the address parameter follows the same format described above in the patient table. Lastly, the phone number for the pharmacy will be contained in the data type varchar(20). In order to present the phone number in a user-friendly format, the database's application will need to structure phone numbers using symbols like dashes, parentheses, and so on, therefore the varchar data type was deemed the most appropriate.

The retail pharmacy table has three relationships that are illustrated in the ERD which will be explained in further detail below. These relationships are a many-to-many relationship with the pharmaceutical company table, a many-to-many relationship with the prescription drug table, and a one-to-many relationship with the prescription order table.

Prescription_Drug Table

Customer requirements for prescription drug data are that each drug has a generic

CST 363: Databases – Project Database Design and Implementation

name and may have a trade name. Additionally, if the drug has a trade name it is unique. In our prescription drug table, these requirements are met by setting the genericName field with a Not Null constraint, while the trade name only has a Unique constraint. Both of these fields are set to a data type of VARCHAR(45), since the field will contain the drug's names. Lastly, since an integer value was used as the primary key for the table since the name fields are expected to either not be unique identifiers or contain null values.

Name	Data Type	Constraints	Description
drugID	INT	Primary Key, Not Null	Primary key for prescription drug table
genericName	VARCHAR(45)	Not Null	Generic name for prescription drug
tradeName	VARCHAR(45)	Unique	Unique trade name for prescription drug

Additional customer requirements associated with the prescription drug table were handled through defined relationships with the other tables in the database. For example, a one-to-many relationship was defined between the pharmaceutical company and prescription drug tables since each drug is manufactured and sold by a pharmaceutical company, but each pharmaceutical company can manufacture multiple prescription drugs.

Additionally, since multiple prescription drugs can be sold by multiple retail pharmacies, this requirement was fulfilled with a many-to-many relationship. This created the association table prescription_drug_has_retail_pharmacy where the primary key consists of the combination of drugID and retailPharmacyID parameters. The prescription drug price at the referenced retail pharmacy is recorded in this table as drugCost since the price of each drug can vary between pharmacies. A data type DECIMAL(7,2) was chosen for this parameter as this would allow prices to be entered up to the cent and include expensive medication. The database application will need to contain code to restrict values to positive non zero between \$0.01 and \$99,999.99.

Lastly, a one-to-many relationship was defined between the prescription drug and prescription order tables since each prescription order contains only one medication, but one medication can be referenced by multiple prescription orders.

Prescription_Order Table

The prescription order table utilizes the unique prescription RX number as its primary key and references the doctor, patient, and prescription drug tables with foreign keys. To meet the customer requirements, the prescription order table has a one-to-many relationship with all three of the aforementioned tables since patients can have multiple prescription orders which can be written by any doctor. Patients can also have

CST 363: Databases – Project Database Design and Implementation

multiple prescription orders. Prescription orders must contain the prescription drug information. The prescription drug itself, is part of a prescription order, connected to a patient's information, is sold from a retail pharmacy, and originates from a pharmaceutical company. It's also implied that a prescription drug can be referenced by multiple prescription orders. As a safety check, the database application should flag or prompt doctors when prescribing medication that has been recently prescribed to the same patient by a different doctor. This will help ensure that the patient is not being over prescribed medication.

Name	Data Type	Constraints	Description
RXNumber	INT	Primary Key, Not Null, Unique	Prescription RX number used as a primary key
prescribeDate	DATETIME	Not Null	Date prescription was written
prescribeQuantity	INT	Not Null	Quantity of medication prescribed
Doctor_SSN	INT	Foreign Key, Not Null	Reference to the doctor that prescribed medication
Patient_SSN	INT	Foreign Key, Not Null	Reference to the patient the prescribed medication is for
drugID	INT	Foreign Key, Not Null	Reference to medication prescribed
genericOK	TINYINT(1)	Not Null	Boolean value (0 or 1) indicating whether medication can be filled with generic formula
fillDate	DATETIME		Date the prescription order was filled
retailPharmacyID	INT		Retail pharmacy that filled the prescription order
pharmaceuticalCompanyID	INT		If prescription order was filled by generic, this field will contain the pharmaceutical company ID that manufactures the generic drug

CST 363: Databases – Project Database Design and Implementation

Pharmaceutical_Company Table

The pharmaceutical company table contains the name and phone number for each prescription drug manufacturer. The name and phone number columns follow the same data type and format used throughout the database for consistency. In order to meet the customer requirements, a many-to-many relationship between the pharmaceutical company and retail pharmacy tables which created the association table named Pharmaceutical_Company_has_Retail_Pharmacy1. This was done since retail pharmacies can have contracts with multiple pharmaceutical companies.

Name	Data Type	Constraints	Description
ID	INT	Primary Key, Not Null, Unique	Primary key used to reference pharmaceutical companies
name	VARCHAR(45)	Not Null	Pharmaceutical company name
phoneNumber	VARCHAR(20)	Not Null	Pharmaceutical company phone number

Additionally, since a pharmaceutical company can produce multiple prescription drugs we implemented a one-to-many relationship between the two tables. Lastly, in cases where a prescription order is filled with a generic drug the pharmaceutical company ID is recorded and referenced within the prescription order table. In order to appropriately characterize this, a one-to-many relationship was created between them without the not null constraint since this parameter is not recorded until the prescription order is filled with a generic medication.

Supervisor Table

Similar to the Specialty Table, this table consists of two columns: ID and Name. The Name column is of type VARCHAR(45) and is meant to contain the name of supervisors. The ID is of type INT and will keep track of all supervisors. The ID is also the table's primary key. Note that the table has a many-to-one relationship with the pharmacy retailers because a supervisor may be working with multiple retailers. Because supervisors may come and go we decided to create a specific table to prevent redundancies and dependencies.

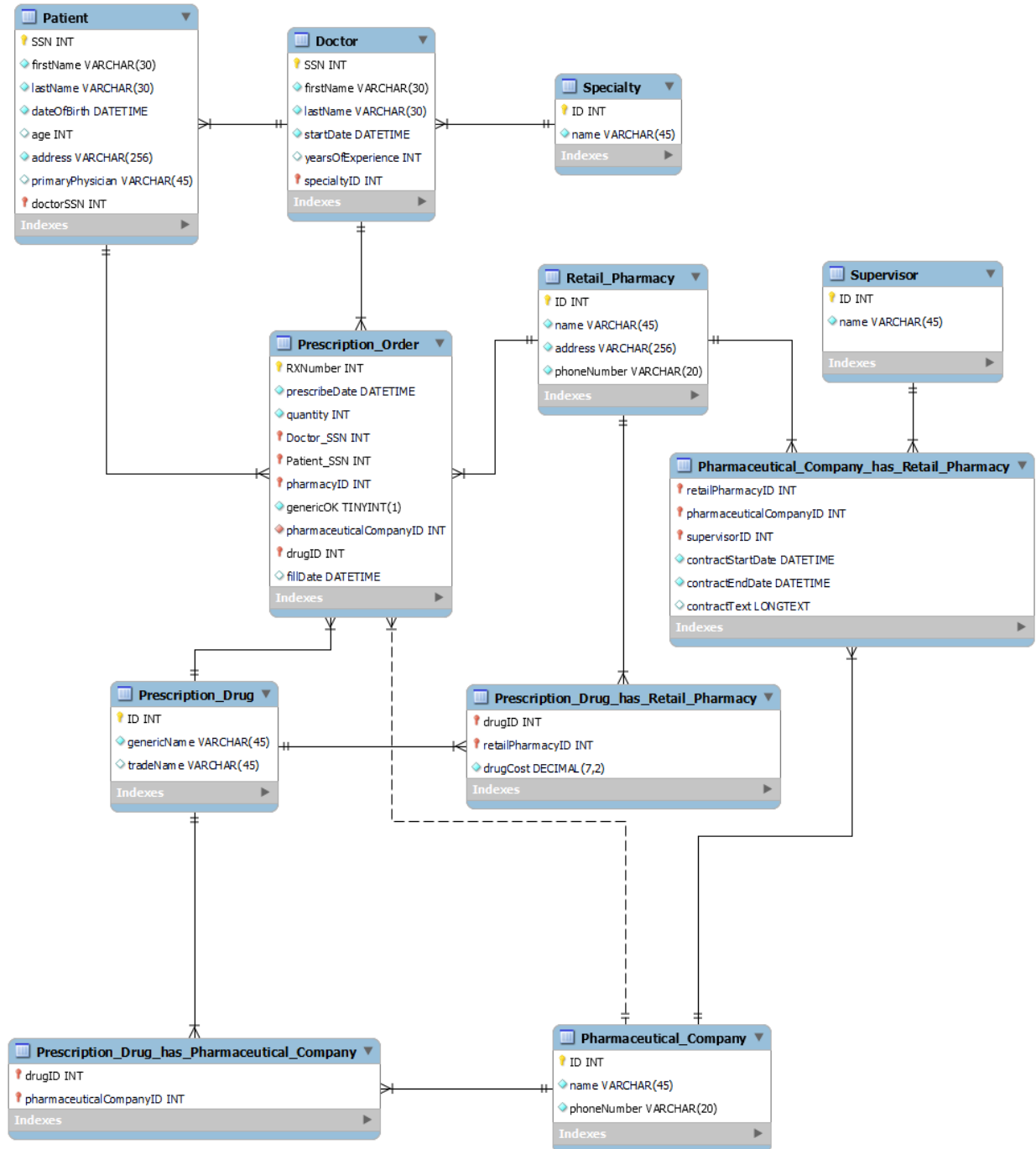
Conclusions

CST 363: Databases – Project Database Design and Implementation

We have put together a database and throughout that process we have learned a lot about what types of issues can arise and how to avoid them. Determining how tables interact with each other is a vital area of the database and we spent a lot of time making sure they were set up correctly per the customer's requirements.

CST 363: Databases – Project Database Design and Implementation

Appendix A: Database Entity-Relationship Diagram



Appendix B: Sample Queries

–What's the most common prescription drug prescribed by Oncology?

```
SELECT pd.genericName, pd.tradeName, COUNT(RXNumber)
FROM prescription_order as po
JOIN doctor AS d ON po.doctor_ssn = d.ssn
JOIN specialty AS s ON d.specialtyID = s.ID
JOIN prescription_drug AS pd ON po.drugID = pd.ID
WHERE s.name = 'Oncology'
GROUP BY 1,2
HAVING COUNT(RXNumber) = (
    SELECT MAX(rx_count)
FROM (
    SELECT COUNT(RXNumber) as rx_count
    FROM prescription_order as a
    JOIN doctor AS b ON a.doctor_SSN = b.SSN
    WHERE b.specialtyID = (SELECT ID FROM specialty WHERE name =
'Oncology')
    GROUP BY drugID
) AS c);
```

–What is the lowest price option available for ibuprophen?

```
SELECT pd.genericName, rtp.name AS retailName
FROM prescription_drug AS pd
JOIN prescription_drug_has_retail_pharmacy AS rt ON rt.drugID = pd.ID
JOIN retail_pharmacy rtp ON rt.retailPharmacyID = rtp.ID
WHERE pd.genericName = 'ibuprophen' AND
```

CST 363: Databases – Project Database Design and Implementation

```
rt.drugCost = (  
SELECT MIN(drugCost)  
FROM prescription_drug_has_retail_pharmacy AS pd_rt  
JOIN prescription_drug AS pd ON pd_rt.drugID = pd.ID  
WHERE pd.genericName = 'ibuprophen');
```

–Which doctor(s) prescribes the most medications?

```
CREATE view all_prescription_orders AS (  
SELECT doctor.SSN AS doctorID, doctor.firstName, doctor.lastName,  
count(prescription_order.RXNumber) AS prescriptionCount  
FROM doctor, prescription_order  
WHERE doctor.SSN = prescription_order.Doctor_SSN  
GROUP BY doctor.SSN);
```

```
SELECT * FROM all_prescription_orders WHERE prescriptionCount = (  
SELECT MAX(prescriptionCount) FROM all_prescription_orders);
```

–What is the total revenue for Paul's Pharmacy between dates 01/01/2022 and 01/01/2023?

```
SELECT pdr.drugCost*COUNT(po.quantity) AS totalRevenue FROM  
prescription_order po  
JOIN prescription_drug pd ON pd.ID = po.drugID  
JOIN prescription_drug_has_retail_pharmacy pdr ON pdr.drugID = pd.ID  
JOIN retail_pharmacy rp ON rp.ID = pdr.retailPharmacyID  
WHERE po.prescribeDate BETWEEN '2022-01-01' AND '2023-01-01' AND rp.name =  
'Paul's Pharmacy'  
GROUP BY pdr.drugCost;
```

–What is the most commonly used manufacturer for Folic Acid generic prescriptions?

```
CREATE VIEW pharma_company_prescriptions AS(  
SELECT pharmaceuticalCompanyID, pc.name, count(RXNumber) AS prescriptionCount
```

CST 363: Databases – Project Database Design and Implementation

```
FROM prescription_order AS po  
JOIN prescription_drug AS pd ON po.drugID = pd.ID  
JOIN Pharmaceutical_Company As pc ON po.pharmaceuticalCompanyID = pc.ID  
WHERE genericName = 'Folic Acid'  
GROUP BY pharmaceuticalCompanyID);
```

```
SELECT * FROM pharma_company_prescriptions  
WHERE prescriptionCount = (SELECT MAX(prescriptionCount) FROM  
pharma_company_prescriptions)
```

CST 363: Databases – Project Database Design and Implementation

Appendix C: Usage Screenshots

New Prescription

The image displays two side-by-side screenshots of the 'New Prescription Form' interface. The left screenshot shows a successful prescription creation. The form fields are filled with: Doctor SSN: 123450700, Doctor First Name: Dr. Scott, Doctor Last Name: Chang, Patient SSN: 123450049, Patient First Name: Kelly, Patient Last Name: Welch, Drug Name: amitriptyline, Generic? (unchecked), and Quantity: 25. A 'Create Prescription' button is at the bottom. The right screenshot shows an error state. The Doctor SSN field is highlighted in blue with the text 'Error: Doctor Information Does Not Match Records.' below it. The other fields are the same as in the left screenshot, but the 'Create Prescription' button is disabled.

- The new prescription is filled out in the form, all information is validated and checked against respective databases to make sure it is filled out correctly. The 'Generic?' checkbox indicates if the patient is ok with receiving the generic drug rather than a trade or brand name, and it is reflected once the patient goes to fill the prescription. The doctor can put either the generic or a trade name in the 'Drug Name' field and this checkbox will still be the deciding factor on what the patient receives.

Fill Prescription

The image displays two side-by-side screenshots of the 'Request Prescription be filled.' interface. The left screenshot shows a successful request. The form fields are filled with: Rx: 473686032, Patient SSN: 123450049, Patient Last Name: Welch, Pharmacy Name: Lafoya's Pharmacy, and Pharmacy Address: 4418 6th Street North, Glendale, CA. A 'Request Fill for Prescription' button is at the bottom. The right screenshot shows an error state. The Rx field is highlighted in blue with the text 'Error: Prescription Already Filled' below it. The other fields are the same as in the left screenshot, but the 'Request Fill for Prescription' button is disabled.

- The request prescription is filled out in the form, all information gets validated and the inputted information is checked against the Prescription_Order table.

CST 363: Databases – Project Database Design and Implementation

Once the prescription is filled it shows the prescription information, including the correct drug name based on the doctor's choice from the 'Generic ?' checkbox. The last portion displays an error message for another, already filled prescription, although there are other error messages for different data mismatches like before.

Register New Patient

The image displays two versions of a 'Register as new user' form side-by-side. The left form is the successful state, and the right form is the error state.

Left Screenshot (Successful Registration):

- Register as new user**
- Your SSN: [masked]
- Your First Name: Pedro
- Your Last Name: Pascal
- Birth Date: 04/02/1975
- Street: 12345 6th st
- City: Austin
- State: TX
- Zipcode: 96748
- Primary Physician Name: Dr. Alex
- [Register](#)
- Registration successful.**
- Patient SSN: 123456578
- First Name: Pedro
- Last Name: Pascal
- Date Of Birth: 1975-04-02
- Age: 48
- Address: 12345 6th st, Austin, TX 96748
- Primary Physician: Dr. Edward Olson
- [Edit](#) | [Main Menu](#)

Right Screenshot (Error State):

- Register as new user**
- Invalid SSN entered. Please re-enter 9 digit SSN
- Your SSN: [empty]
- Your First Name: Pedro
- Your Last Name: Pascal
- Birth Date: 04/02/1975
- Street: 12345 6th St
- City: Austin
- State: TX
- Zipcode: 95223
- Primary Physician Name: Dr. Edward Olson
- [Register](#)

- The registration page prompts the user to enter all the required information for a new patient. Once entered the data is validated and show to the user to confirm the successful entry into the database. There are various error messages for invalid data (Displayed on the Right).

CST 363: Databases – Project Database Design and Implementation

Search and Update Existing Patient

Enter patient SSN and name

Patient SSN:


Patient Last Name:

Patient SSN: 123450588
First Name: Marcus
Last Name: Blanchard
Date Of Birth: 1958-05-16
Age: 65
Address: 1218 Vine Street, Houma, MI 14250
Primary Physican: Dr. Erik


[Edit](#) | [Main Menu](#)

Update Patient Profile

SSN:

First Name: 

Last Name:

Birth Date: 

Age:

Address:

Primary Physician Name:

Update successful

Patient SSN: 123450588
First Name: Marcus
Last Name: Blanchard
Date Of Birth: 1958-05-16
Age: 65
Address: 212 Pearl Street, Lansing, VT 66045
Primary Physican: Dr. Courtney

[Edit](#) | [Main Menu](#)

- The user searches for a patient based on SSN and Last Name and the database returns the information to be viewed. The user can then edit the patient's information, which is then sent back to the database and once again displayed once the update is completed.

CST 363: Databases – Project Database Design and Implementation

Pharmacy Manager and FDA Data Requests

<pre>Please input PharmacyID (Enter To Continue). 07 Enter Start Date as yyyy-MM-dd: 1900-01-01 Enter End Date as yyyy-MM-dd: 2024-01-01 Rachel's Pharmacy Quantity of Prescription Drugs filled between 1900-01-01 and 2024-01-01 GENERIC NAME TRADE NAME TOTAL QUANTITY FILLED metoprolol succinate XL Toprol 55 spironolactone Aldactone 143 azithromycin Zithromax 93 levofloxacin Levaquin 144 fluoxetine HCl Prozac 66</pre>	<pre>Enter Generic Drug Name (Hit Enter To Skip): acetaminophen and codeine Enter Trade Drug Name (Hit Enter To Skip): Enter Start Date as yyyy/mm/dd: 1900/01/01 Enter End Date as yyyy/mm/dd: 2024/01/01 Name: Dr. Douglas Jennings Amount Prescribed: 144 Name: Dr. Chelsea Dickerson Amount Prescribed: 200</pre>
--	---

- The Pharmacy Manager request (Left) lets a user input a pharmacyID, startDate, and endDate, and returns the information of the imputed pharmacy and the quantity of prescription drugs filled between the two inputted dates.
- The FDA request (Right) has the user select a drug by both generic and trade names, enter start and end dates, and is then returned the doctor's name and the quantity of drugs that each doctor has prescribed.

Appendix C: Relational Schema (SQL Code)

-- MySQL Script generated by MySQL Workbench

-- Mon Jan 30 13:34:07 2023

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET GLOBAL FOREIGN_KEY_CHECKS=0;

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_SUBSTI
TUTION';

-- Schema cst363

CREATE SCHEMA IF NOT EXISTS `cst363` DEFAULT CHARACTER SET utf8 ;

USE `cst363` ;

-- Table `Specialty`

CREATE TABLE IF NOT EXISTS `Specialty` (
 `ID` INT NOT NULL,
 `name` VARCHAR(45) NOT NULL,
 PRIMARY KEY (`ID`),
 UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE)
ENGINE = InnoDB;

CST 363: Databases – Project Database Design and Implementation

```
-- -----  
-- Table `Doctor`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `Doctor` (  
  `SSN` INT NOT NULL,  
  `firstName` VARCHAR(30) NOT NULL,  
  `lastName` VARCHAR(30) NOT NULL,  
  `startDate` DATETIME NOT NULL,  
  `yearsOfExperience` INT NULL,  
  `specialtyID` INT NOT NULL,  
  PRIMARY KEY (`SSN`, `specialtyID`),  
  UNIQUE INDEX `SSN_UNIQUE` (`SSN` ASC) VISIBLE,  
  INDEX `fk_Doctor_Specialty1_idx` (`specialtyID` ASC) VISIBLE,  
  CONSTRAINT `fk_Doctor_Specialty1`  
    FOREIGN KEY (`specialtyID`)  
      REFERENCES `Specialty` (`ID`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `Patient`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `Patient` (  
  `SSN` INT NOT NULL,
```

CST 363: Databases – Project Database Design and Implementation

```
`firstName` VARCHAR(30) NOT NULL,  
`lastName` VARCHAR(30) NOT NULL,  
`dateOfBirth` DATETIME NOT NULL,  
`age` INT NULL,  
`address` VARCHAR(256) NOT NULL,  
`primaryPhysician` VARCHAR(45) NULL,  
`doctorSSN` INT NOT NULL,  
PRIMARY KEY (`SSN`, `doctorSSN`),  
INDEX `fk_Patient_Doctor1_idx` (`doctorSSN` ASC) VISIBLE,  
UNIQUE INDEX `SSN_UNIQUE` (`SSN` ASC) VISIBLE,  
CONSTRAINT `fk_Patient_Doctor1`  
FOREIGN KEY (`doctorSSN`)  
REFERENCES `Doctor` (`SSN`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `Pharmaceutical_Company`  
-- -----
```

```
CREATE TABLE IF NOT EXISTS `Pharmaceutical_Company` (  
  `ID` INT NOT NULL,  
  `name` VARCHAR(45) NOT NULL,  
  `phoneNumber` VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`ID`),  
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE)  
ENGINE = InnoDB;
```

CST 363: Databases – Project Database Design and Implementation

```
-- -----  
-- Table `Retail_Pharmacy`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `Retail_Pharmacy` (  
  `ID` INT NOT NULL,  
  `name` VARCHAR(45) NOT NULL,  
  `address` VARCHAR(256) NOT NULL,  
  `phoneNumber` VARCHAR(20) NOT NULL,  
  PRIMARY KEY (`ID`),  
  UNIQUE INDEX `pharmacyID_UNIQUE` (`ID` ASC) VISIBLE)  
ENGINE = InnoDB;
```

```
-- -----  
-- Table `Prescription_Drug`  
-- -----  
  
CREATE TABLE IF NOT EXISTS `Prescription_Drug` (  
  `ID` INT NOT NULL,  
  `genericName` VARCHAR(45) NOT NULL,  
  `tradeName` VARCHAR(45) NULL,  
  PRIMARY KEY (`ID`),  
  UNIQUE INDEX `tradeName_UNIQUE` (`tradeName` ASC) VISIBLE,  
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE)  
ENGINE = InnoDB;
```

CST 363: Databases – Project Database Design and Implementation

-- Table `Doctor_has_Prescription_Order`

```
CREATE TABLE IF NOT EXISTS `Doctor_has_Prescription_Order` (  
  `doctorSSN` INT NOT NULL,  
  `RXNumber` INT NOT NULL,  
  PRIMARY KEY (`doctorSSN`, `RXNumber`),  
  INDEX `fk_Doctor_has_Prescription_Order_Doctor1_idx` (`doctorSSN` ASC)  
  VISIBLE,  
  CONSTRAINT `fk_Doctor_has_Prescription_Order_Doctor1`  
    FOREIGN KEY (`doctorSSN`)  
      REFERENCES `Doctor` (`SSN`)  
      ON DELETE NO ACTION  
      ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

-- Table `Patient_has_Prescription_Order`

```
CREATE TABLE IF NOT EXISTS `Patient_has_Prescription_Order` (  
  `patientSSN` INT NOT NULL,  
  `RXNumber` INT NOT NULL,  
  PRIMARY KEY (`patientSSN`, `RXNumber`),  
  INDEX `fk_Patient_has_Prescription_Order_Patient1_idx` (`patientSSN` ASC)  
  VISIBLE,  
  CONSTRAINT `fk_Patient_has_Prescription_Order_Patient1`  
    FOREIGN KEY (`patientSSN`)  
      REFERENCES `Patient` (`SSN`)
```

CST 363: Databases – Project Database Design and Implementation

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- Table `Prescription_Order_has_Prescription_Drug`

CREATE TABLE IF NOT EXISTS `Prescription_Order_has_Prescription_Drug` (

 `RXNumber` INT NOT NULL,

 `ID` INT NOT NULL,

 PRIMARY KEY (`RXNumber`, `ID`),

 INDEX `fk_Prescription_Order_has_Prescription_Drug_Prescription_Dr_idx`
 (`ID` ASC) VISIBLE,

 CONSTRAINT

 `fk_Prescription_Order_has_Prescription_Drug_Prescription_Drug1`

 FOREIGN KEY (`ID`)

 REFERENCES `Prescription_Drug` (`ID`)

 ON DELETE NO ACTION

 ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- Table `Supervisor`

CREATE TABLE IF NOT EXISTS `Supervisor` (

 `ID` INT NOT NULL,

 `name` VARCHAR(45) NOT NULL,

CST 363: Databases – Project Database Design and Implementation

```
PRIMARY KEY (`ID`),  
UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE)  
ENGINE = InnoDB;  
  
-----  
-- Table `Pharmaceutical_Company_has_Retail_Pharmacy`  
-----  
  
CREATE TABLE IF NOT EXISTS  
`Pharmaceutical_Company_has_Retail_Pharmacy` (  
  `retailPharmacyID` INT NOT NULL,  
  `pharmaceuticalCompanyID` INT NOT NULL,  
  `supervisorID` INT NOT NULL,  
  `contractStartDate` DATETIME NOT NULL,  
  `contractEndDate` DATETIME NOT NULL,  
  `contractText` LONGTEXT NULL,  
  PRIMARY KEY (`retailPharmacyID`, `pharmaceuticalCompanyID`,  
  `supervisorID`),  
  INDEX  
  `fk_Pharmaceutical_Company_has_Retail_Pharmacy1_Retail_Pharm_idx`  
  (`retailPharmacyID` ASC) VISIBLE,  
  INDEX `fk_Pharmaceutical_Company_has_Retail_Pharmacy1_Supervisor1_idx`  
  (`supervisorID` ASC) VISIBLE,  
  INDEX  
  `fk_Pharmaceutical_Company_has_Retail_Pharmacy1_Pharmaceutic_idx`  
  (`pharmaceuticalCompanyID` ASC) VISIBLE,  
  CONSTRAINT  
  `fk_Pharmaceutical_Company_has_Retail_Pharmacy1_Retail_Pharmacy1`  
  FOREIGN KEY (`retailPharmacyID`)  
  REFERENCES `Retail_Pharmacy` (`ID`)  
  ON DELETE NO ACTION
```

CST 363: Databases – Project Database Design and Implementation

ON UPDATE NO ACTION,

CONSTRAINT

`fk_Pharmaceutical_Company_has_Retail_Pharmacy1_Supervisor1`

FOREIGN KEY (`supervisorID`)

REFERENCES `Supervisor` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT

`fk_Pharmaceutical_Company_has_Retail_Pharmacy1_Pharmaceutical1`

FOREIGN KEY (`pharmaceuticalCompanyID`)

REFERENCES `Pharmaceutical_Company` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- Table `Prescription_Drug_has_Pharmaceutical_Company`

CREATE TABLE IF NOT EXISTS

`Prescription_Drug_has_Pharmaceutical_Company` (

`drugID` INT NOT NULL,

`pharmaceuticalCompanyID` INT NOT NULL,

PRIMARY KEY (`drugID`, `pharmaceuticalCompanyID`),

INDEX

`fk_Prescription_Drug_has_Pharmaceutical_Company_Pharmaceuti_idx`
(`pharmaceuticalCompanyID` ASC) VISIBLE,

INDEX `fk_Prescription_Drug_has_Pharmaceutical_Company_Prescriptio_idx`
(`drugID` ASC) VISIBLE,

CONSTRAINT

`fk_Prescription_Drug_has_Pharmaceutical_Company_Prescription_1`

CST 363: Databases – Project Database Design and Implementation

```
FOREIGN KEY (`drugID`)
REFERENCES `Prescription_Drug` (`ID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT
`fk_Prescription_Drug_has_Pharmaceutical_Company_Pharmaceutica1`
FOREIGN KEY (`pharmaceuticalCompanyID`)
REFERENCES `Pharmaceutical_Company` (`ID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

-----
-- Table `Prescription_Drug_has_Retail_Pharmacy`
-----

CREATE TABLE IF NOT EXISTS `Prescription_Drug_has_Retail_Pharmacy` (
  `drugID` INT NOT NULL,
  `retailPharmacyID` INT NOT NULL,
  `drugCost` DECIMAL(7,2) NOT NULL,
  PRIMARY KEY (`drugID`, `retailPharmacyID`),
  INDEX `fk_Prescription_Drug_has_Retail_Pharmacy_Retail_Pharmacy1_idx`
  (`retailPharmacyID` ASC) VISIBLE,
  INDEX `fk_Prescription_Drug_has_Retail_Pharmacy_Prescription_Drug1_idx`
  (`drugID` ASC) VISIBLE,
  CONSTRAINT
  `fk_Prescription_Drug_has_Retail_Pharmacy_Prescription_Drug1`
  FOREIGN KEY (`drugID`)
  REFERENCES `Prescription_Drug` (`ID`)
  ON DELETE NO ACTION
```

CST 363: Databases – Project Database Design and Implementation

```
ON UPDATE NO ACTION,  
CONSTRAINT `fk_Prescription_Drug_has_Retail_Pharmacy_Retail_Pharmacy1`  
FOREIGN KEY (`retailPharmacyID`)  
REFERENCES `Retail_Pharmacy` (`ID`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
-- Table `Prescription_Order`  
-----
```

```
CREATE TABLE IF NOT EXISTS `Prescription_Order` (  
  `RXNumber` INT NOT NULL,  
  `prescribeDate` DATETIME NOT NULL,  
  `quantity` INT NOT NULL,  
  `Doctor_SSN` INT NOT NULL,  
  `Patient_SSN` INT NOT NULL,  
  `pharmacyID` INT NOT NULL,  
  `genericOK` TINYINT(1) NOT NULL,  
  `pharmaceuticalCompanyID` INT NOT NULL,  
  `drugID` INT NOT NULL,  
  `fillDate` DATETIME NULL,  
  PRIMARY KEY (`RXNumber`, `Doctor_SSN`, `Patient_SSN`, `pharmacyID`,  
  `drugID`),  
  INDEX `fk_Prescription_Order_Doctor1_idx` (`Doctor_SSN` ASC) VISIBLE,  
  INDEX `fk_Prescription_Order_Patient1_idx` (`Patient_SSN` ASC) VISIBLE,  
  INDEX `fk_Prescription_Order_Retail_Pharmacy1_idx` (`pharmacyID` ASC)  
  VISIBLE,
```

CST 363: Databases – Project Database Design and Implementation

**INDEX `fk_Prescription_Order_Prescription_Drug1_idx` (`drugID` ASC)
INVISIBLE,**

**INDEX `fk_Prescription_Order_Pharmaceutical_Company1_idx`
(`pharmaceuticalCompanyID` ASC) VISIBLE,**

CONSTRAINT `fk_Prescription_Order_Doctor1`

FOREIGN KEY (`Doctor_SSN`)

REFERENCES `Doctor` (`SSN`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk_Prescription_Order_Patient1`

FOREIGN KEY (`Patient_SSN`)

REFERENCES `Patient` (`SSN`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk_Prescription_Order_Retail_Pharmacy1`

FOREIGN KEY (`pharmacyID`)

REFERENCES `Retail_Pharmacy` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk_Prescription_Order_Prescription_Drug1`

FOREIGN KEY (`drugID`)

REFERENCES `Prescription_Drug` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk_Prescription_Order_Pharmaceutical_Company1`

FOREIGN KEY (`pharmaceuticalCompanyID`)

REFERENCES `Pharmaceutical_Company` (`ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

CST 363: Databases – Project Database Design and Implementation

ENGINE = InnoDB;

SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

-- What's the most common prescription drug prescribed by Oncology?

SELECT pd.genericName, pd.tradeName, COUNT(RXNumber)

FROM prescription_order as po

JOIN doctor AS d ON po.doctor_ssn = d.ssn

JOIN specialty AS s ON d.specialtyID = s.ID

JOIN prescription_drug AS pd ON po.drugID = pd.ID

WHERE s.name = 'Oncology'

GROUP BY 1,2

HAVING COUNT(RXNumber) = (

SELECT MAX(rx_count)

FROM (

SELECT COUNT(RXNumber) as rx_count

FROM prescription_order as a

JOIN doctor AS b ON a.doctor_SSN = b.SSN

**WHERE b.specialtyID = (SELECT ID FROM specialty WHERE name
= 'Oncology')**

GROUP BY drugID

) AS c);

-- What is the lowest price option available for ibuprophen?

SELECT pd.genericName, rtp.name AS retailName

FROM prescription_drug AS pd

CST 363: Databases – Project Database Design and Implementation

```
JOIN prescription_drug_has_retail_pharmacy AS rt ON rt.drugID = pd.ID
```

```
JOIN retail_pharmacy rtp ON rt.retailPharmacyID = rtp.ID
```

```
WHERE pd.genericName = 'ibuprophen' AND
```

```
rt.drugCost = (
```

```
SELECT MIN(drugCost)
```

```
FROM prescription_drug_has_retail_pharmacy AS pd_rt
```

```
JOIN prescription_drug AS pd ON pd_rt.drugID = pd.ID
```

```
WHERE pd.genericName = 'ibuprophen');
```

-- Which doctor(s) prescribes the most medications?

```
CREATE view all_prescription_orders AS (
```

```
SELECT doctor.SSN AS doctorID, doctor.firstName, doctor.lastName,  
count(prescription_order.RXNumber) AS prescriptionCount
```

```
FROM doctor, prescription_order
```

```
WHERE doctor.SSN = prescription_order.Doctor_SSN
```

```
GROUP BY doctor.SSN);
```

```
SELECT * FROM all_prescription_orders WHERE prescriptionCount = (
```

```
SELECT MAX(prescriptionCount) FROM all_prescription_orders);
```

-- What is the total revenue for Paul's Pharmacy between dates 01/01/2022 and 01/01/2023?

```
SELECT pdr.drugCost*COUNT(po.quantity) AS totalRevenue FROM  
prescription_order po
```

```
JOIN prescription_drug pd ON pd.ID = po.drugID
```

```
JOIN prescription_drug_has_retail_pharmacy pdr ON pdr.drugID = pd.ID
```

```
JOIN retail_pharmacy rp ON rp.ID = pdr.retailPharmacyID
```

```
WHERE po.prescribeDate BETWEEN '2022-01-01' AND '2023-01-01' AND rp.name  
= 'Paul's Pharmacy'
```

```
GROUP BY pdr.drugCost;
```

CST 363: Databases – Project Database Design and Implementation

-- What is the most commonly used manufacturer for Folic Acid generic prescriptions?

```
CREATE VIEW pharma_company_prescriptions AS(  
SELECT pharmaceuticalCompanyID, pc.name, count(RXNumber) AS  
prescriptionCount  
FROM prescription_order AS po  
JOIN prescription_drug AS pd ON po.drugID = pd.ID  
JOIN Pharmaceutical_Company As pc ON po.pharmaceuticalCompanyID = pc.ID  
WHERE genericName = 'Folic Acid'  
GROUP BY pharmaceuticalCompanyID);  
  
SELECT * FROM pharma_company_prescriptions  
WHERE prescriptionCount = (SELECT MAX(prescriptionCount) FROM  
pharma_company_prescriptions)
```