BayTech Meds

Project 1 Database Design
CST 363 Introduction to Database Systems

January 25, 2022

Jennah Yoasin & Yukio Rivera

Introduction

PURPOSE

This relational database design serves as a tool for storing information for several pharmacies. Data ranges from information about specific patients, specific doctors and their specialties and years of experience, as well as information about the drugs themselves; like their generic names and trade names, and finally prescriptions. Following this, data of the pharmaceutical companies themselves and what contracts they are signed with pharmacies. Ultimately, the purpose of this relational database is to be a source of data that can be used to query for specific data about these topics and the results of the queries can be essentially used for scientific or relevant research purposes.

SPECIFIC REQUIREMENTS

SQL DATABASE:

A database is considered to be relational when there are connections and constraints between specific tables of data. Here is a list of the requirements that were applied in order to develop this relational database through the relationships between tables:

- **Patient**: Must contain information about the patient's:
  - Social Security Number (SSN)
  - Full Name
  - Age
  - Address
  - Primary Doctor Name
    - **RELATIONSHIPS:**
      - Every patient can have one doctor/physician
      - Every patient can be prescribed one or more prescriptions.
- **Doctor/Physician**: Must contain information about the doctor's:
  - ID
  - Social Security Number (SSN)
  - Full Name
  - Specialty
  - Years of Experience
    - **RELATIONSHIP**:

            - Any doctor can write a prescription for any patient

- **PharmecuticalCompany**: Must contain information about the company's:
    - Company ID
    - Name
    - Phone Number
        - **RELATIONSHIP:**
            - Company's have contracts with at least one pharmacy.
- **Drug (Medication)**: Must contain information about the drug's:
    - Drug ID
    - Trade Name
    - Generic Formula
        - **RELATIONSHIPS**:
            - Every medication is sold at one or more pharmaceutical companies.
            - Trade name distinguishes between the same drugs sold at different companies.
- **Pharmacy**: Must contain information about the pharmacy's:
    - ID
    - Name
    - Address
    - Phone Number
        - **RELATIONSHIP**:
            - Pharmacy's have contracts with many, at least one, pharmaceutical company.
- **Prescription**: Must contain information about the prescription's:
    - Unique RX Number
    - Prescribed Date
    - Quantity
    - Filled Prescription Date
    - Cost
        - **RELATIONSHIP:**
            - **UNIQUE** RX number is for every single **one** drug, **one** patient and is prescribed by **one** doctor.
- **Contract**: Must contain information about the contract's:
    - Contract ID
    - Start Date
    - End Date
    - Text
    - Price
        - **RELATIONSHIP**:

- Each contract is appointed by a supervisor from a company.

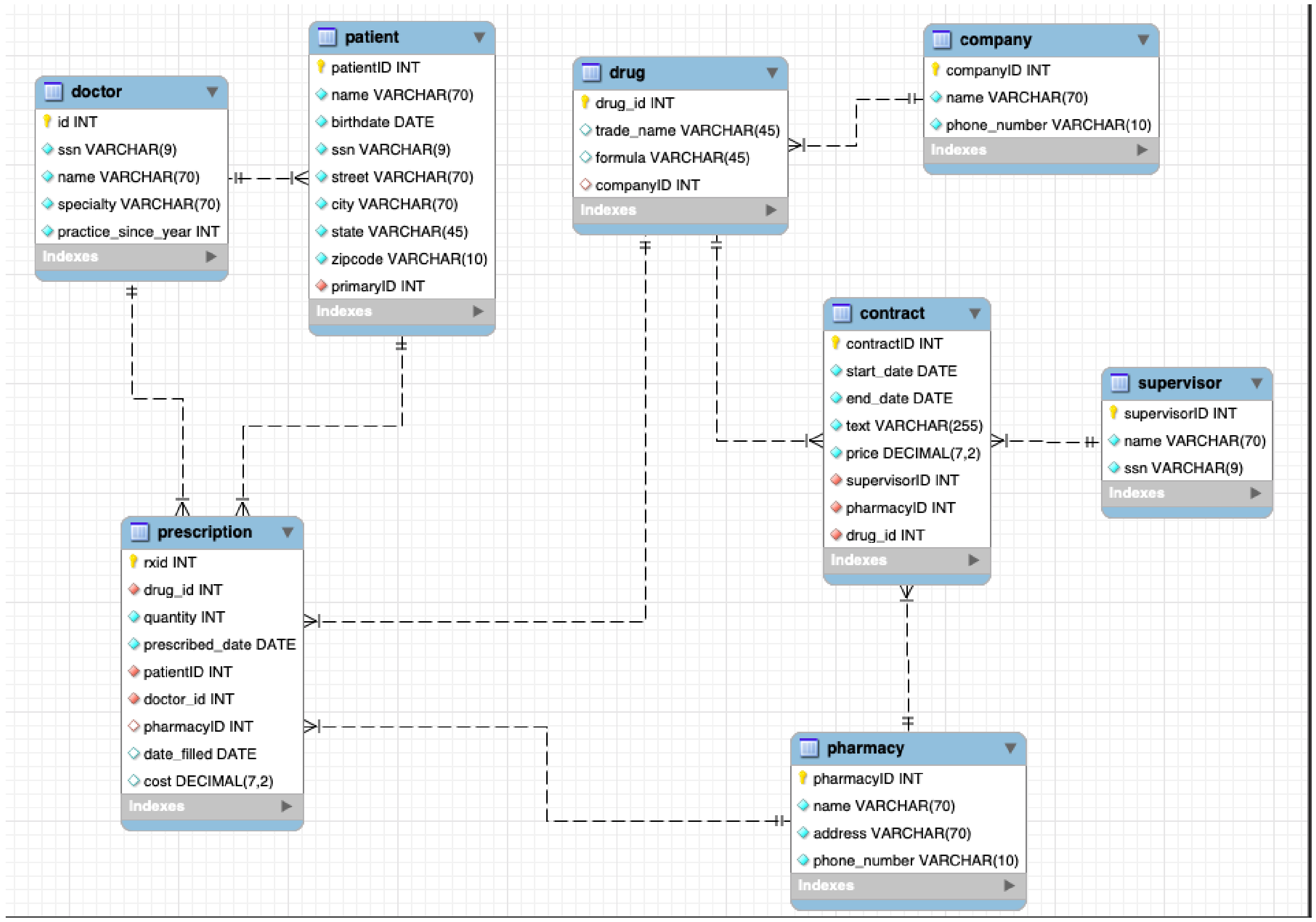
❖ **Supervisor**: Must contain information about the supervisor's name:

- ➢ Supervisor ID
- ➢ Full Name
- ➢ SSN

■ **RELATIONSHIP**:

- Each supervisor can supervise one or more contracts.

ER MODEL:



The way the ER Model is structured:

➔ Each relationship is shown using the Crow's foot notation concept.

- ⧫ The constraints between the tables represents the direct relationships between the tables
- ⧫ Every table has a relationship to at least one other table
  - As for the requirements of the database listed above the ER diagram, every single required relationship is shown in this ER chart. For example, the relationship between patient and doctor:

        - A doctor can have multiple patients, but a patient can only have one primary doctor. As shown, the patient gains the foreign key of the doctor's SSN(which is the primary key for the doctor's table) and that is how they are related. Every other relationship is similarly shown simply.

- ➔ Primary & Foreign Keys
    - ⧫ Referring to the requirements, each table is connected to another through primary and foreign links
        - The primary keys are shown as the yellow key symbols, and the foregin keys are shown as the red diamond symbols on the ER diagram above.
        - It is simple to follow through the links using these keys. For example:
            - Drug and Company are linked through the company's ID. Therefore, the primary key of the company table is companyID and the foreign key of the drug table is also the companyID. This is how all of the relationships of this database are shown on the ER diagram.
- ➔ Every column of every row was created with a datatype, depending on the data it is expected to hold.
    - ⧫ Using our common knowledge, we decided that for phone numbers, id's, RX numbers, experience levels and ages, we would use integers.
    - ⧫ For names, formulas, addresses, and doctor's specialties we decided to use VARCHAR's of a various number limit, depending on the information being added to each column.
    - ⧫ Finally, we also used the DATE datatype, for obviously, the prescription's FilledDate, PrescribedDate, StartDate, and EndDate
- ➔ Unique & Null Datatypes:
    - ⧫ Data that is made UNIQUE:
        - According to the database's requirements, the RX Numbers of the drugs must be unique, so we made sure to do that.
        - Understanding what Social Security Numbers and what Phone Numbers are, we decided to make those values also unique
        - The ID's like the pharmacyID, contractid, and supervisor's ID were also made unique to avoid confusion.
    - ⧫ Data that is made NULL:
        - The only null value we decided to make was the FilledPharmacyID and the prescriptions FilledDate
            - This is because if a patient's prescription was not picked up from the pharmacy, there is no date.

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_SUBSTITUTION';
-- -----------------------------------------------------
-- Schema pharmacy
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `pharmacy` DEFAULT CHARACTER SET utf8 ;
USE `pharmacy` ;
-- -----------------------------------------------------
-- Table `pharmacy`.`doctor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`doctor` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `ssn` VARCHAR(9) NOT NULL,
  `name` VARCHAR(70) NOT NULL,
  `specialty` VARCHAR(70) NOT NULL,
  `practice_since_year` INT NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pharmacy`.`patient`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`patient` (
  `patientID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(70) NOT NULL,
  `birthdate` DATE NOT NULL,
  `ssn` VARCHAR(9) NOT NULL,
  `street` VARCHAR(70) NOT NULL,
  `city` VARCHAR(70) NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `zipcode` VARCHAR(10) NOT NULL,
  `primaryID` INT NOT NULL,
  PRIMARY KEY (`patientID`),
  INDEX `fk_patient_doctor_idx` (`primaryID` ASC) VISIBLE,
  CONSTRAINT `fk_patient_doctor`
    FOREIGN KEY (`primaryID`)
    REFERENCES `pharmacy`.`doctor` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `pharmacy`.`company`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`company` (
  `companyID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(70) NOT NULL,
  `phone_number` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`companyID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pharmacy`.`drug`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`drug` (
  `drug_id` INT(11) NOT NULL AUTO_INCREMENT,
  `trade_name` VARCHAR(45) NULL,
  `formula` VARCHAR(45) NULL,
  `companyID` INT NULL,
  PRIMARY KEY (`drug_id`),
  INDEX `fk_drug_company1_idx` (`companyID` ASC) VISIBLE,
  CONSTRAINT `fk_drug_company1`
    FOREIGN KEY (`companyID`)
    REFERENCES `pharmacy`.`company` (`companyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pharmacy`.`pharmacy`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`pharmacy` (
  `pharmacyID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(70) NOT NULL,
  `address` VARCHAR(70) NOT NULL,
  `phone_number` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`pharmacyID`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `pharmacy`.`prescription`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`prescription` (
 `rxid` INT NOT NULL AUTO_INCREMENT,
 `drug_id` INT(11) NOT NULL,
 `quantity` INT NOT NULL,
 `prescribed_date` DATE NOT NULL,
 `patientID` INT NOT NULL,
 `doctor_id` INT NOT NULL,
 `pharmacyID` INT NULL,
 `date_filled` DATE NULL,
 `cost` DECIMAL(7,2) NULL,
 PRIMARY KEY (`rxid`),
 INDEX `fk_prescription_drug1_idx` (`drug_id` ASC) VISIBLE,
 INDEX `fk_prescription_patient1_idx` (`patientID` ASC) VISIBLE,
 INDEX `fk_prescription_doctor1_idx` (`doctor_id` ASC) VISIBLE,
 INDEX `fk_prescription_pharmacy1_idx` (`pharmacyID` ASC) VISIBLE,
 CONSTRAINT `fk_prescription_drug1`
  FOREIGN KEY (`drug_id`)
  REFERENCES `pharmacy`.`drug` (`drug_id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_prescription_patient1`
  FOREIGN KEY (`patientID`)
  REFERENCES `pharmacy`.`patient` (`patientID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_prescription_doctor1`
  FOREIGN KEY (`doctor_id`)
  REFERENCES `pharmacy`.`doctor` (`id`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_prescription_pharmacy1`
  FOREIGN KEY (`pharmacyID`)
  REFERENCES `pharmacy`.`pharmacy` (`pharmacyID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `pharmacy`.`supervisor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`supervisor` (
  `supervisorID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(70) NOT NULL,
  `ssn` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`supervisorID`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `pharmacy`.`contract`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `pharmacy`.`contract` (
  `contractID` INT NOT NULL AUTO_INCREMENT,
  `start_date` DATE NOT NULL,
  `end_date` DATE NOT NULL,
  `text` VARCHAR(255) NOT NULL,
  `price` DECIMAL(7,2) NOT NULL,
  `supervisorID` INT NOT NULL,
  `pharmacyID` INT NOT NULL,
  `drug_id` INT(11) NOT NULL,
  PRIMARY KEY (`contractID`),
  INDEX `fk_contract_supervisor1_idx` (`supervisorID` ASC) VISIBLE,
  INDEX `fk_contract_pharmacy1_idx` (`pharmacyID` ASC) VISIBLE,
  INDEX `fk_contract_drug1_idx` (`drug_id` ASC) VISIBLE,
  CONSTRAINT `fk_contract_supervisor1`
    FOREIGN KEY (`supervisorID`)
    REFERENCES `pharmacy`.`supervisor` (`supervisorID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_contract_pharmacy1`
    FOREIGN KEY (`pharmacyID`)
    REFERENCES `pharmacy`.`pharmacy` (`pharmacyID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_contract_drug1`
    FOREIGN KEY (`drug_id`)
    REFERENCES `pharmacy`.`drug` (`drug_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

select ssn from doctor;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
-- PHARMACY INSERT INTO STATEMENTS --
INSERT INTO pharmacy (name, address, phone_number) VALUES ('CVS on 41st', '231 41st Ave', '5551111111');
INSERT INTO pharmacy (name, address, phone_number) VALUES  ('Downtown CVS', '11 Downtown Street',
'5552222222');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Rite Aid on 41st', '233 41st Ave', '5553333333');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Downtown Rite Aid', '13 Downtown Street',
'5554444444');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Walgreens on 41st', '235 41st Ave', '5555555555');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Downtown Walgreens', '15 Downtown Street',
'5556666666');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Walmart Pharmacy', '300 Walmart Ave',
'5557777777');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Costco Pharmacy', '9 Downtown Street',
'5558888888');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Amazon Medicine', '123 Online Ave',
'5559999999');
INSERT INTO pharmacy (name, address, phone_number) VALUES ('Safeway Pharmacy', '12 Other Road',
'5550000000');
-- DRUG INSERT INTO STATEMENTS --
INSERT INTO drug (drug_id, trade_name, formula) VALUES
(1,'Tylenol with Codeine','acetaminophen and codeine'),
(2,'Proair Proventil','albuterol aerosol'),
(3,'Accuneb','albuterol HFA'),
(4,'Fosamax','alendronate'),
(5,'Zyloprim','allopurinol'),
(6,'Xanax','alprazolam'),
(7,'Elavil','amitriptyline'),
(8,'Augmentin','amoxicillin and clavulanate K+'),
(9,'Amoxil','amoxicillin'),
(10,'Adderall XR','amphetamine and dextroamphetamine XR'),
(11,'Tenormin','atenolol'),
(12,'Lipitor','atorvastatin'),
(13,'Zithromax','azithromycin'),
(14,'Lotrel','benazepril and amlodipine'),
(15,'Soma','carisoprodol'),
(16,'Coreg','carvedilol'),
(17,'Omnicef','cefdinir'),
(18,'Celebrex','celecoxib'),
(19,'Keflex','cephalexin'),
(20,'Cipro','ciprofloxacin'),
(21,'Celexa','citalopram'),
(22,'Klonopin','clonazepam'),
(23,'Catapres','clonidine HCl'),
(24,'Plavix','clopidogrel'),
(25,'Premarin','conjugated estrogens'),
(26,'Flexeril','cyclobenzaprine'),
(27,'Valium','diazepam'),
(28,'Voltaren','diclofenac sodium'),
(29,'Yaz','drospirenone and ethinyl estradiol'),
(30,'Cymbalta','Duloxetine'),
(31,'Vibramycin','doxycycline hyclate'),
(32,'Vasotec','enalapril'),
(33,'Lexapro','escitalopram'),
(34,'Nexium','esomeprazole'),
(35,'Zetia','ezetimibe'),
(36,'Tricor','fenofibrate'),
(37,'Allegra','fexofenadine'),
(38,'Diflucan','fluconozole'),
(39,'Prozac','fluoxetine HCl'),
(40,'Advair','fluticasone and salmeterol inhaler'),
(41,'Flonase','fluticasone nasal spray'),
(42,'Folic Acid','folic acid'),
(43,'Lasix','furosemide'),
(44,'Neurontin','gabapentin'),
(45,'Amaryl','glimepiride'),
(46,'Diabeta','glyburide'),
(47,'Glucotrol','glipizide'),
(48,'Microzide','hydrochlorothiazide'),
(49,'Lortab','hydrocodone and acetaminophen'),
(50,'Motrin','ibuprophen'),
(51,'Lantus','insulin glargine'),
(52,'Imdur','isosorbide mononitrate'),
(53,'Prevacid','lansoprazole'),
(54,'Levaquin','levofloxacin'),
(55,'Levoxl','levothyroxine sodium'),
(56,'Zestoretic','lisinopril and hydrochlorothiazide'),
(57,'Prinivil','lisinopril'),
(58,'Ativan','lorazepam'),
(59,'Cozaar','losartan'),
(60,'Mevacor','lovastatin'),
(61,'Mobic','meloxicam'),
(62,'Glucophage','metformin HCl'),
(63,'Medrol','methylprednisone'),
(64,'Toprol','metoprolol succinate XL'),
(65,'Lopressor','metoprolol tartrate'),
(66,'Nasonex','mometasone'),
(67,'Singulair','montelukast'),
(68,'Naprosyn','naproxen'),
(69,'Prilosec','omeprazole'),
(70,'Percocet','oxycodone and acetaminophen'),
(71,'Protonix','pantoprazole'),
(72,'Paxil','paroxetine'),
(73,'Actos','pioglitazone'),
(74,'Klor-Con','potassium Chloride'),
(75,'Pravachol','pravastatin'),
(76,'Deltasone','prednisone'),
(77,'Lyrica','pregabalin'),
(78,'Phenergan','promethazine'),
(79,'Seroquel','quetiapine'),
(80,'Zantac','ranitidine'),
(81,'Crestor','rosuvastatin'),
(82,'Zoloft','sertraline HCl'),
(83,'Viagra','sildenafil HCl'),
(84,'Vytorin','simvastatin and ezetimibe'),
(85,'Zocor','simvastatin'),
(86,'Aldactone','spironolactone'),
(87,'Bactrim DS','sulfamethoxazole and trimethoprim DS'),
(88,'Flomax','tamsulosin'),
(89,'Restoril','temezepam'),
(90,'Topamax','topiramate'),
(91,'Ultram','tramadol'),
(92,'Aristocort','triamcinolone Ace topical'),
(93,'Desyrel','trazodone HCl'),
(94,'Dyazide','triamterene and hydrochlorothiazide'),
(95,'Valtrex','valaciclovir'),
(96,'Diovan','valsartan'),
(97,'Effexor XR','venlafaxine XR'),
(98,'Calan SR','verapamil SR'),
(99,'Ambien','zolpidem');
```

- Considering normalization, it was taken into account to make sure every table had its specific information solely, rather than combining the same types of data of different tables together. One specific place where rearranging took place was normalizing the supervisor table. We initially had it combined with the contract table but then realized that multiple supervisors can have multiple contracts with different pharmacies and this would produce invalid information when queried. We also decided to spit up the addresses for patients into separate columns, instead of combining the entire address into just 1 column.

Queries :

1) Display the pharmacy id of pharmacies that sell drugs with the trade name of 'Accuneb';

SQL STATEMENT:

```
SELECT DISTINCT prescription.pharmacyID from prescription
INNER JOIN drug ON prescription.drug_id=drug.drug_id
WHERE trade_name = 'Accuneb';
```

2) Display the patient name as PatientName, medication number, amount of meds, date, prescribing doctor as DoctorName, and pharmacy name as PharmacyName the meds need to be picked up at.

SQL STATEMENT:

```
SELECT p.name AS PatientName, pr.rxid, pr.patientID,
        pr.prescribed_date, d.name AS DoctorName, ph.name AS PharmacyName
FROM patient p
INNER JOIN doctor d ON p.primaryID=d.id
INNER JOIN prescription pr ON d.id=pr.doctor_id
INNER JOIN pharmacy ph ON pr.pharmacyID=ph.pharmacyID;
```

3) Using a sub-query, find the average cost of the drugs in all pharmacies and display the maximum of that average cost and the medication trade name.

```
SELECT MAX(AverageCost), trade_name
FROM (SELECT AVG(cost) AS AverageCost, trade_name
FROM prescription p
INNER JOIN drug d ON p.drug_id = d.drug_id
GROUP BY trade_name) AS AvgCost
GROUP BY trade_name;
```

4)Display the names and SSNs of all patients and doctors that have a ssn number that ends with the number 9. Add the prefix Dr. before doctor names as FullName. USE UNION.

SQL STATEMENT:

```
SELECT CONCAT('Dr.', d.name) as FullName, d.ssn
FROM doctor d
WHERE ssn LIKE '%9'
UNION
SELECT p.name, p.ssn
FROM patient p
WHERE ssn LIKE '%9';
```

5)Display the sum of the patients each doctor has as PatientCount. Order by the doctor's name.

SQL STATEMENT:

```
SELECT d.name, COUNT(p.primaryID) as PatientCount
FROM doctor d
INNER JOIN patient p ON d.id=p.primaryID
GROUP BY d.name
ORDER BY d.name;
```

Java Spring Project in Eclipse

For this project, we were also responsible for creating a Java spring project in Eclipse in order to implement both a java application and a web application from our database created previously. There were specific requirements set for each section:

Java Application:

For the Java Application requirement, we were required to complete the DataGenerate program. The DataGenerate program basically worked for inserting information into the Pharmacy database tables(that we created prior). For each table, there was specific data that had to be inputted and there were conditions required to cover for each column. For example, a social security number with 9 digits can't start with 0 or 9, have middle digits of 00, or have the last 4 digits as 0000. We covered all of the other requirements as well to ensure we generated the data for the tables correctly. For the second java application requirement, we had a few options to choose from. We decided to create the ManagerReport application. ManagerReport worked as a Java application that ultimately produces information from the database's table when queried for it. More specifically, it asks a manager(the user) for the pharmacyID and the range for the filled prescriptions to list all of the medication names and quantities for that specified pharmacy in the pharmacyID.

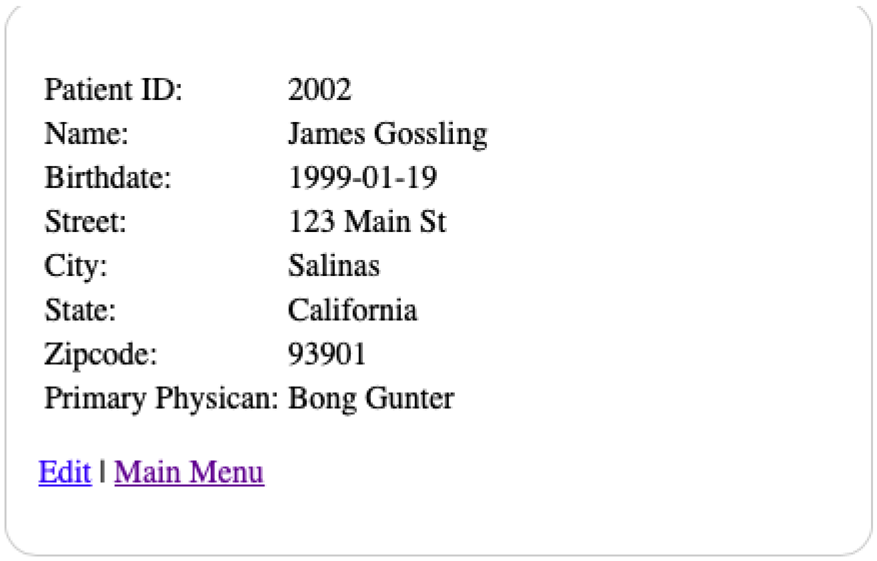
Web Application:

For the web application, we were expected to fill in missing pieces of the provided Java classes. As the project description explains, the Spring Tool in Eclipse uses a Model-View-Controller pattern. With this, the classes that controlled the web application, are the controller classes. That is where all of our work took place. We were expected to complete two of the listed options under the web application specifications. We ended up choosing to work with the controller patient class. The ControllerPatient.java class basically set up the code web application for the menu options of "Register as a new patient" and "Display patient data". For registering a new patient, we decided to first check for invalid user input for each section of data. After all of the input is validated, we coded in the insert statements to ensure that we were also adding a patient to the database itself and not just on the web application. From here, once all of the user's input is inserted into the patient table, the patient is successfully added to the database and their information can be viewed.

For the second part of the requirement, the viewing of a patient's information must also allow the user to edit their address and primary doctor information. So, for this we worked in the updatePatient function in order to allow the user to update their information and have it also update the database and be viewed in the Java application.

SCREENSHOTS

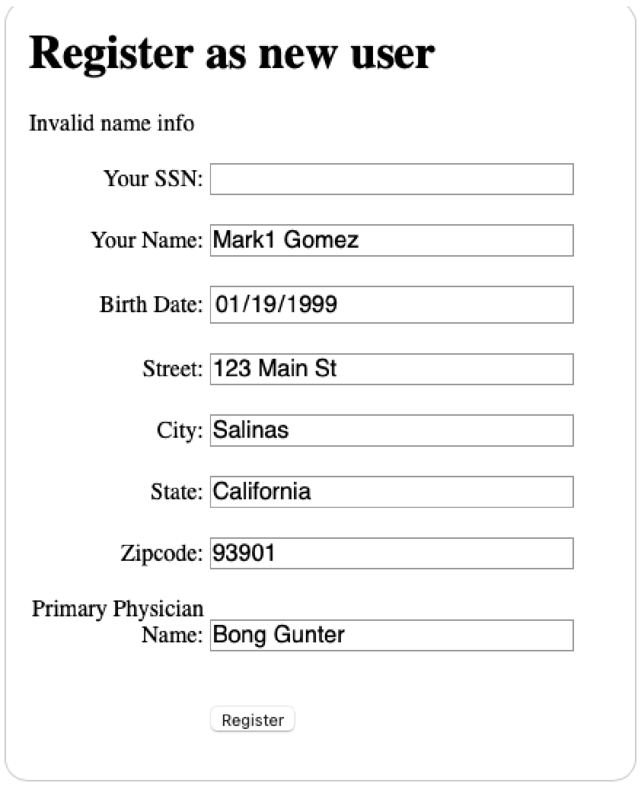
Registering a patient (successfully):



Patient ID:	2002
Name:	James Gossling
Birthdate:	1999-01-19
Street:	123 Main St
City:	Salinas
State:	California
Zipcode:	93901
Primary Physican:	Bong Gunter

Edit | Main Menu

Registering a patient with invalid name info:

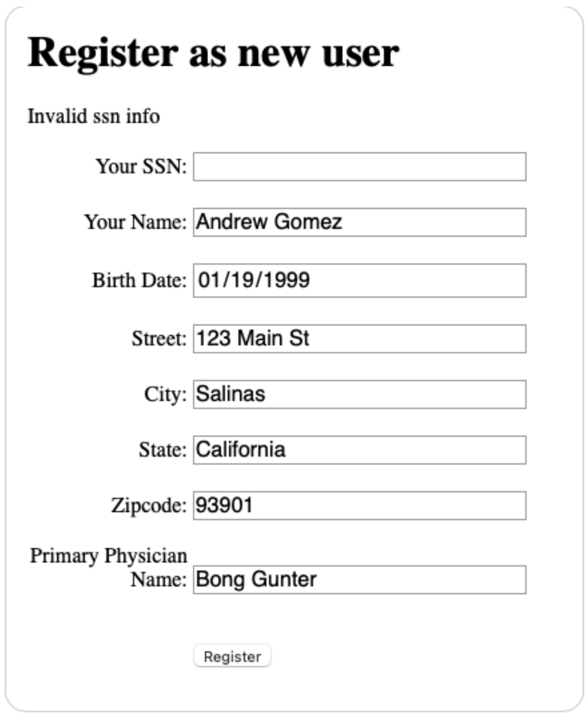


Register as new user

Invalid name info

Your SSN:	
Your Name:	Mark1 Gomez
Birth Date:	01/19/1999
Street:	123 Main St
City:	Salinas
State:	California
Zipcode:	93901
Primary Physician Name:	Bong Gunter

Registering a patient with invalid ssn info:

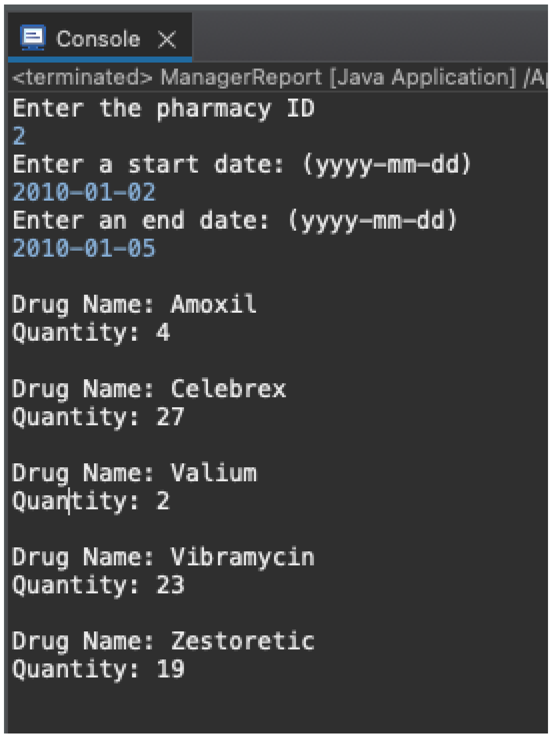


Register as new user

Invalid ssn info

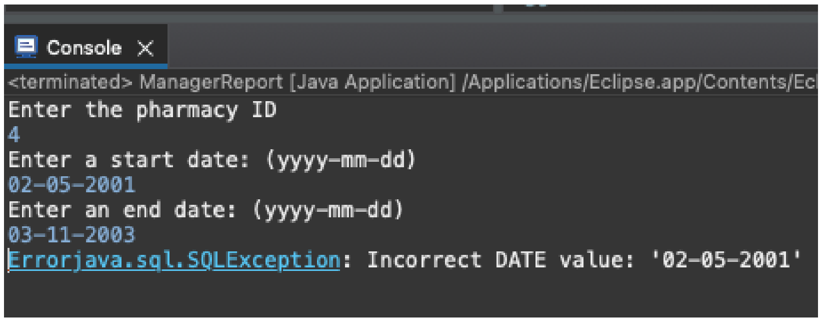
Your SSN:	
Your Name:	Andrew Gomez
Birth Date:	01/19/1999
Street:	123 Main St
City:	Salinas
State:	California
Zipcode:	93901
Primary Physician Name:	Bong Gunter

Successful run of ManagerReport.java



```
Console ×
<terminated> ManagerReport [Java Application] /A
Enter the pharmacy ID
2
Enter a start date: (yyyy-mm-dd)
2010-01-02
Enter an end date: (yyyy-mm-dd)
2010-01-05

Drug Name: Amoxil
Quantity: 4

Drug Name: Celebrex
Quantity: 27

Drug Name: Valium
Quantity: 2

Drug Name: Vibramycin
Quantity: 23

Drug Name: Zestoretic
Quantity: 19
```

Unsuccessful run of MangerReport.java(invalid date):



```
Console ×
<terminated> ManagerReport [Java Application] /Applications/Eclipse.app/Contents/Ec
Enter the pharmacy ID
4
Enter a start date: (yyyy-mm-dd)
02-05-2001
Enter an end date: (yyyy-mm-dd)
03-11-2003
Errorjava.sql.SQLException: Incorrect DATE value: '02-05-2001'
```

CONCLUSION

The creation of this database took plenty of thought and understanding of the way databases are structured and how they work. We started off by thinking about how we would split off the information into tables. We gathered the information for each potential table and decided on which tables will stand alone and which portions of data can possibly be combined into one table. By starting off like this, we were able to influence the idea of having a normalized database. After setting up each of the tables with their data, we had to think about how we would link the tables together so that the database becomes relational. By doing this, we were able to connect and figure out how certain tables had to be correlated as requested by the requirements of the project. This project was definitely beneficial in giving us a better idea of working for a real job as the directions were somewhat vague and left us open for our own assumptions and creativity. Having that amount of freedom of thinking was helpful and felt relieving so that we can put our own thoughts into the assignment. Overall, the implementation of this database was an encouraging experience and gave us both a better understanding of the way databases are generated.