
Pharmacy Database Project

Prepared by:

**Zora Li
Haomiao Shi
Sherry Zhao**

Nov. 5th, 2021

1. Application Background

Pharmacies are retail shops that provide pharmaceutical medicine for customers. Most pharmacies are located in commercial areas, such as inside grocery stores. Pharmacies sell both over-the-counter drugs, medications that can be bought without a doctor's order, and prescription drugs, medications that require doctor prescriptions and can only be consumed by the prescribed customer. Going to pharmacies is typically the most common way for people to get their prescriptions filled. Pharmacy employees are able to give health-related suggestions or answer questions on prescription drugs.

The pharmacy system possesses a large amount of data. There exist multiple entities that interact with each other. People go to see doctors at the hospital and receive prescriptions. Their prescriptions are filled by employees working at the pharmacy. Customers can also purchase nonprescription drugs at the pharmacy. Drug manufacturers are contacted when pharmacies have shortages of certain medicines so that drugs can be stocked in a timely manner. Therefore, it is vital to implement a database for the pharmacy system so that all the information involved can be properly tracked and monitored.

Pharmacy Database helps to manage the pharmacy system by recording the data and trajectories of various drugs. A pharmacy records the kinds, number, and other basic information of medicines stocked from the providers. When a customer comes to the pharmacy and purchases medicine, the information of the trade, including the information of the customer, the employee who sells the medicine, and the date of the trade will be recorded, thus greatly reducing the possibility of illegal drug trade. Always filling one's prescriptions at the same local pharmacy also helps the pharmacy to keep a record of all drugs that have been purchased by the same customer, preventing drug interactions, which are reactions between multiple drugs where their pharmacological effect is altered. With the aid of Pharmacy Database, even smaller pharmacies would be able to keep up with their large-scale competition.

2. Data Description

Our database has seven entities and nine relations. Customer, Employee, and Doctor contain personal information about customers who go to pharmacies, employees who work at pharmacies, and doctors who prescribe medications for customers. Medicine, Pharmacy and Manufacturer hold information about drugs that are provided by pharmacies, pharmacies that sell nonprescription and prescription medicines, and

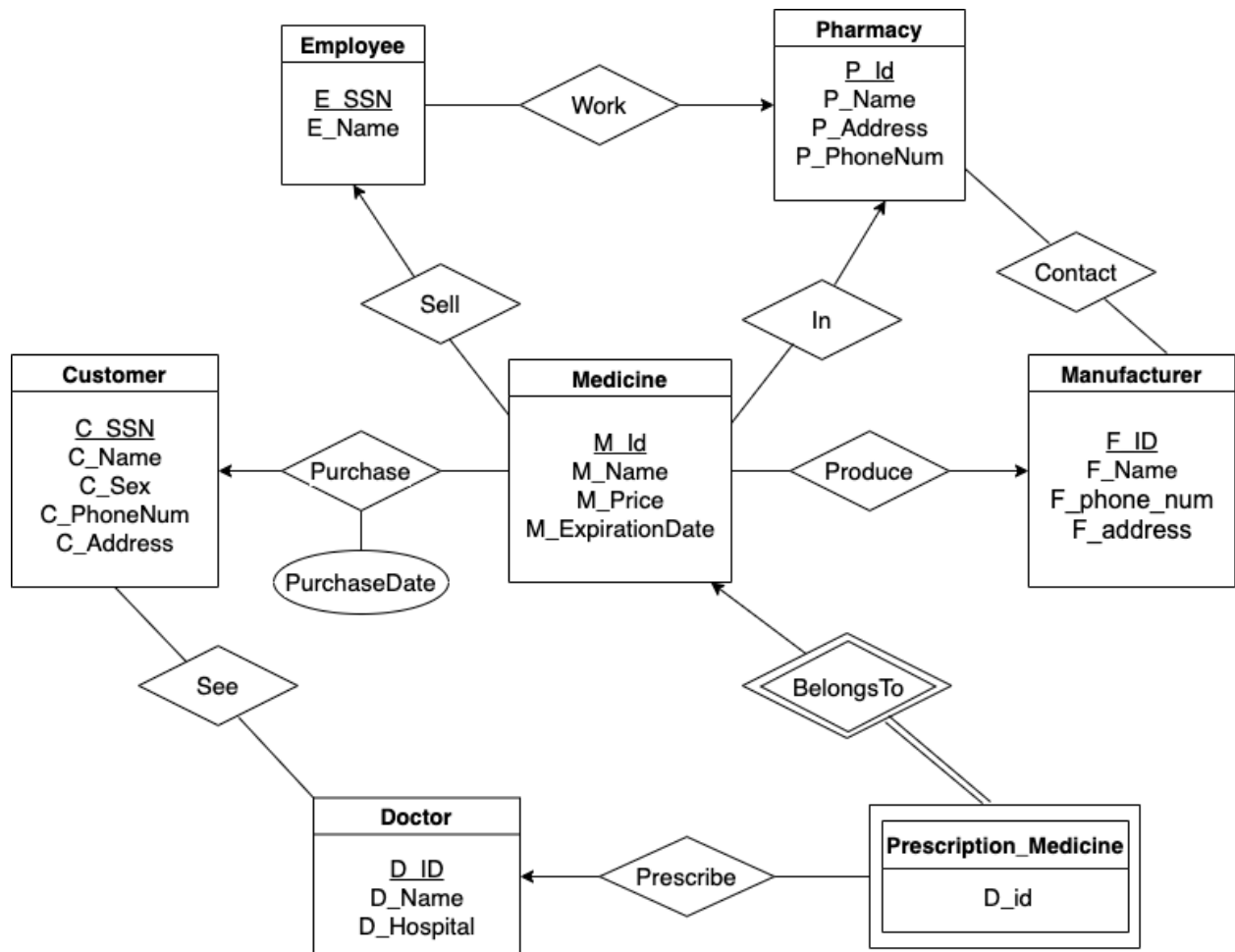
manufacturers that produce drugs for pharmacies. Prescription Medicine is a weak entity that belongs to the strong entity Medicine and is merged with the relation Prescribe.

A pharmacy may have multiple employees while each employee can only work at one pharmacy. Every employee may sell multiple bottles of drugs, but each bottle can only be sold by one employee. Each pharmacy may have multiple numbers of different drugs in stock while each bottle of medicine can only be in one pharmacy. Manufacturers produce different drugs, and each bottle of drug can only be produced by one manufacturer. Customers can purchase multiple bottles of medicines, and each bottle of drugs can only be bought by one customer. A doctor can prescribe multiple medications while each medication can only be prescribed by one doctor. The relation See records the customer who goes to see the doctor that prescribes medicine for him or her. A customer can go to different doctors, and each doctor can see multiple patients. The relation Contact tracks the pharmacy that contacts the drug manufacturer that produces medicine. A pharmacy can contact multiple manufacturers, and a manufacturer can be contacted by multiple pharmacies to stock their medicine storage.

Our database possesses foreign key constraints, NOT NULL constraints, and UNIQUE constraints. Primary keys, including employee SSN, pharmacy ID, medicine ID, doctor ID, manufacturer ID, and customer SSN, need to follow the NOT NULL and UNIQUE constraints. Foreign keys are used to demonstrate how tables are connected to each other, and every foreign key is the primary key of another table to which this table is connected. Thus, the entities and their relationships illustrated in the ER diagram can be well-preserved. Besides the primary keys, many other attributes also follow the NOT NULL constraints. For instance, the P_id in the Employee table cannot be null so that the pharmacy the employee works at can be properly recorded.

All of our tables are in BCNF as shown in the following sections below, thus decreasing the amount of possible redundancies in the database. The data currently used in our database is randomly generated using Mockaroo.

3. ER Diagram



4. Functional Dependencies, Normalizations

(Relational Schema is described in the next section)

Employee:

The attributes of the employee contain information about the individual employee. The information of one employee is different from the other employee. Hence, the primary key

E_SSN can determine all other attributes in this relation schema.

- $E_SSN \rightarrow \{E_Name, P_Id\}$

This is a BCNF relation because E_SSN is a super key.

Pharmacy:

The attributes of the pharmacy store information about the individual pharmacy. The information of one pharmacy is different from the other pharmacy. Like Employee, P_Id is the only key to trace all attributes in this relation schema.

- $P_Id \rightarrow \{P_Name, P_Address\}$

This is a BCNF relation because P_Id is a super key.

Manufacturer :

The attributes of the manufacturer indicate information about the individual manufacturer. The information of one manufacturer is different from the other manufacturer. Like Employee, F_Id is the only key to trace all attributes in this relation schema.

- $F_Id \rightarrow \{F_Name, F_phone_num, F_Address\}$

This is a BCNF relation because F_Id is a super key.

Customer:

A sample constraint is “No two customers can have the same name and phone number” which means the combination of name and phone number can uniquely identify each customer. Note that $\{C_Name, C_phoneNum\}$ is a superkey since its proper set is not a super key (C_Name is not a super key, $C_phoneNum$ is not a super key). Then we have the following functional dependencies:

- $\{C_Name, C_phoneNum\} \rightarrow \{C_SSN\}$
- $\{C_SSN\} \rightarrow \{C_Name, C_Sex, C_phoneNum, C_Address\}$

By using the transitive law from ArmStrong's axiom

- $\{C_Name, C_phoneNum\} \rightarrow \{C_Name, C_Sex, C_phoneNum, C_Address\}$

This is a BCNF relation because $\{C_Name, C_phoneNum\}, \{C_SSN\}$ are super keys.

Doctor:

A sample constraint is “No two doctors can have the same name” which means the doctor's name can uniquely identify each doctor. Then we have the following functional dependencies:

- $\{D_Name\} \rightarrow \{D_ID\}$
- $\{D_ID\} \rightarrow \{D_Name, D_Hospital\}$

By using the transitive law from ArmStrong's axiom

- $\{D_Name\} \rightarrow \{D_Name, D_Hospital\}$

This is a BCNF relation because $\{D_Name\}, \{D_ID\}$ are super keys.

Medicine:

The attributes of the medicine include information about each medicine. A medicine name or a medicine id can not uniquely identify a kind of medicine, but no two medicines can have the same name and same id. $\{M_ID, M_Name\}$ is the super key to determine all attributes in this relation schema.

- $\{M_ID, M_Name\} \rightarrow \{M_Price, M_Expiration_Date, E_SSN, P_Id, F_ID, C_SSN, Purchase_date\}$

This is a BCNF relation because $\{M_ID, M_Name\}$ is a super key.

Prescription_Medicine:

Prescription medicine is a weak entity that relies on the medicine table. A prescription medicine does not have a primary key to trace all attributes, but each prescription medicine needs to be prescribed by a doctor. Hence, {D_ID} is the super key to find all attributes in this relation schema.

- $\{D_ID\} \rightarrow \{M_ID\}$

This is a BCNF relation because {D_ID} is a super key.

Contact, See:

Both of these relationships are many-to-many relationships. For Contact, A pharmacy can contact many manufacturers to produce medicine. Similarly, a manufacturer can be contacted by many pharmacies. For See, a doctor can see many customers to prescribe prescription medicines. Similarly, a customer can see multiple doctors in one day. In both relationships, none of its attributes can be a key. Hence, none of these relationships have any functional dependencies.

5. Relational Schema

Entities:

```
create table employee(  
  E_SSN INT NOT NULL UNIQUE ,  
  E_Name VARCHAR(50),  
  P_Id INT NOT NULL,  
  FOREIGN KEY (P_Id) REFERENCES pharmacy (P_Id),  
  CONSTRAINT pk_employee PRIMARY KEY (E_SSN)  
);
```

```
create table pharmacy(  
  P_Id INT NOT NULL UNIQUE,  
  P_Name VARCHAR(100) NOT NULL,  
  P_Address VARCHAR(100) NOT NULL,  
  CONSTRAINT pk_pharmacy PRIMARY KEY (P_Id)  
);
```

```
create table medicine(  
  M_Id INT NOT NULL UNIQUE,  
  M_Name VARCHAR(100) NOT NULL,  
  M_Price INT NOT NULL ,
```

```
M_Expiration_Date DATE NOT NULL ,
E_SSN INT,
P_Id INT NOT NULL,
F_ID INT NOT NULL,
C_SSN INT,
Purchase_date DATE,
FOREIGN KEY (P_Id) REFERENCES pharmacy (P_Id),
FOREIGN KEY (E_SSN) REFERENCES employee (E_SSN),
FOREIGN KEY (F_ID) REFERENCES manufacturer (F_ID),
FOREIGN KEY (C_SSN) REFERENCES customer (C_SSN),
CONSTRAINT pk_medicine PRIMARY KEY (M_Id,M_Name)
);
```

```
create table doctor(
  D_ID INT NOT NULL UNIQUE,
  D_Name VARCHAR(50) NOT NULL,
  D_Hospital VARCHAR(100) NOT NULL,
  CONSTRAINT pk_doctor PRIMARY KEY (D_ID)
);
```

```
create table manufacturer (
  F_ID INT NOT NULL UNIQUE,
  F_Name VARCHAR(100),
  F_phone_num VARCHAR(10),
  F_address VARCHAR,
  PRIMARY KEY (F_ID)
);
```

```
create table customer (
  C_SSN INT NOT NULL UNIQUE,
  C_Name VARCHAR(50),
  C_Sex VARCHAR(10),
  C_PhoneNum VARCHAR(10),
  C_Address VARCHAR(50),
  PRIMARY KEY (C_SSN)
);
```

```
create table prescription_medicine(
  M_id INT PRIMARY KEY,
```

```

D_id INT,
FOREIGN KEY (M_id) REFERENCES medicine (M_id),
FOREIGN KEY (D_id) REFERENCES doctor (D_id)
);

```

Relationships:

```

create table see(
    D_id INT NOT NULL,
    C_SSN INT NOT NULL,
    FOREIGN KEY (D_id) REFERENCES doctor (D_id),
    FOREIGN KEY (C_SSN) REFERENCES customer (C_SSN),
    PRIMARY KEY (D_id, C_SSN)
);

```

```

create table contact(
    P_id INT NOT NULL,
    F_ID INT NOT NULL,
    FOREIGN KEY (P_id) REFERENCES pharmacy (P_id),
    FOREIGN KEY (F_ID) REFERENCES manufacturer (F_ID),
    PRIMARY KEY (P_id, F_ID)
);

```

6. Example queries

Query 1: This query is used to find the purchase history of medicines. It gives the purchase data, medicine name, employee name, pharmacy name, price, and doctor name. The result table can be used to trace the trades of medicines. The query full outer joins medicine, prescription medicine and doctor, and natural joins employee, customer and pharmacy to get the purchase history table.



```

SELECT m.purchase_date, m.m_name, employee.e_name, customer.c_name, pharmacy.p_name, m.m_price, m.d_name
FROM ((medicine FULL OUTER JOIN prescription_medicine ON medicine.m_id = prescription_medicine.m_id) AS med
FULL OUTER JOIN doctor ON med.d_id = doctor.d_id) AS m,
employee, customer, pharmacy
WHERE employee.e_ssn = m.e_ssn AND customer.c_ssn = m.c_ssn AND pharmacy.p_id = m.p_id

```

$$\Pi_{m.purchase_date, m.m_name, employee.e_name, customer.c_name, pharmacy.p_name, m.price, m.d_name}(\rho_m((medicine \bowtie prescription_medicine) \bowtie doctor) \bowtie employee \bowtie customer \bowtie pharmacy)$$

Query 1 result:

	Data Output	Explain	Messages	Notifications			
	 purchase_date date	 m_name character varying (100)	 e_name character varying (50)	 c_name character varying (50)	 p_name character varying (100)	 m_price integer	 d_name character varying (50)
1	2020-11-01	Luminal	Skipp Kealey	Joejoe	Exactech, Inc.	6	Ellie Peterson
2	2020-11-09	Luminal	Paolina Bergeon	Maria	Exactech, Inc.	6	Ellie Peterson
3	2020-11-09	Luminal	Paolina Bergeon	Maria	Exactech, Inc.	6	Gregory House
4	2018-01-05	Valium	Redd Meuse	Alex	eGain Corporation	8	John Moss
5	2018-02-04	Valium	Willow Godsafe	Alex	eGain Corporation	8	Lea Young
6	2019-04-01	Valium	Willow Godsafe	Maddy	eGain Corporation	8	Gregory House
7	2019-11-09	Vicodin	Gweneth Mayhow	Alex	Vertex Energy, Inc	4	Gregory House
8	2019-11-09	Vicodin	Ludvig Peebles	Alex	Vertex Energy, Inc	4	Gregory House
9	2019-04-01	Demerol	Ludvig Peebles	Maddy	Vertex Energy, Inc	9	John Moss
10	2019-04-11	Demerol	Gweneth Mayhow	Steve	Vertex Energy, Inc	9	Lea Young
11	2020-11-01	Ritalin	Paolina Bergeon	Joejoe	Nuveen Senior Income Fund	7	Gregory House
12	2020-11-02	Ritalin	Ludvig Peebles	Joejoe	Vertex Energy, Inc	7	Gregory House
13	2020-11-09	Xanax	Cornelius Mooreed	Maria	Helios and Matheson Analytics Inc	3	Ellie Peterson
14	2021-01-31	Olanzapine	Zelma Critchell	Steve	Nuveen Senior Income Fund	91	[null]
15	2021-09-23	OMNIPAQUE	Ludvig Peebles	Alex	Vertex Energy, Inc	23	[null]
16	2021-11-22	Prednisone	Ruthy Rodger	Steve	Nuveen Senior Income Fund	20	[null]
17	2021-05-23	TremorSoothe	Ludvig Peebles	Alex	Vertex Energy, Inc	79	[null]
18	2021-04-07	ESIKA 3-IN-1 PRO MAKE UP FOUNDATION SPF ...	Gweneth Mayhow	Alex	Vertex Energy, Inc	69	[null]
19	2021-03-04	Diclofenac Sodium	Ruthy Rodger	Steve	Nuveen Senior Income Fund	83	[null]
20	2021-11-24	Ultravist	Skipp Kealey	Joejoe	Exactech, Inc.	40	[null]
21	2021-02-18	Topco	Cornelius Mooreed	Maddy	Helios and Matheson Analytics Inc	13	[null]
22	2021-01-26	HYZAAR	Jany Ruddlesden	Maddy	Helios and Matheson Analytics Inc	83	[null]
23	2021-05-13	CVS	Redd Meuse	Maria	eGain Corporation	87	[null]
24	2021-09-21	Clonidine Hydrochloride	Zelma Critchell	Steve	Nuveen Senior Income Fund	20	[null]
25	2021-09-25	Quinapril	Willow Godsafe	Maria	eGain Corporation	54	[null]
26	2021-10-10	Aricept	Redd Meuse	Maria	eGain Corporation	59	[null]
27	2021-02-16	good sense cold head congestion	Cornelius Mooreed	Maddy	Helios and Matheson Analytics Inc	86	[null]
28	2021-08-28	Hydrocortisone	Paolina Bergeon	Joejoe	Exactech, Inc.	84	[null]
29	2021-09-17	VECURONIUM BROMIDE	Skipp Kealey	Joejoe	Exactech, Inc.	59	[null]
30	2021-01-11	BUPROPION HYDROCHLORIDE	Paolina Bergeon	Joejoe	Exactech, Inc.	63	[null]

Query 2: This query finds the medicines stored in each pharmacy and their minimum price. It gives the medicine name, pharmacy name, number of medicine in storage, and minimum price. This query is used for medicine storage management. The user can learn what medicines are sold in which pharmacies and compare the minimum prices.

```
SELECT medicine.m_name, pharmacy.p_name, count(medicine.m_id), min(medicine.m_price)
FROM medicine, pharmacy
WHERE medicine.p_id = pharmacy.p_id and medicine.c_ssn IS NULL
GROUP BY medicine.m_name, pharmacy.p_name
```

$$medicine.m_name, pharmacy.p_name \mathcal{G}_{count(medicine.m_id), min(medicine.m_price)}(\sigma_{medicine.c_ssn \neq null}(medicine \bowtie pharmacy))$$

Query 2 result:

	m_name character varying (100)	p_name character varying (100)	count bigint	min integer
1	Demerol	Helios and Matheson Analytics Inc	3	9
2	Ritalin	eGain Corporation	12	7
3	Ritalin	Nuveen Senior Income Fund	4	7
4	Valium	Helios and Matheson Analytics Inc	2	8
5	Vicodin	Helios and Matheson Analytics Inc	3	4
6	Luminal	Vertex Energy, Inc	1	6
7	Clonidine Hydrochloride	Nuveen Senior Income Fund	2	20
8	CVS	eGain Corporation	2	87
9	Olanzapine	Nuveen Senior Income Fund	2	91
10	Xanax	Helios and Matheson Analytics Inc	2	3
11	Luminal	Exactech, Inc.	5	6

Query 3: This query finds the name of the pharmacy which sells all kinds of medicine. If a customer wants to buy many kinds of medicines, the customer can use the query to find a pharmacy with a full range of medicines so that the customer does not need to go to several places to buy all the medicines needed. The subqueries in this clause finds if the pharmacy appears in the same tuple with all kinds of medicines. If the result of subqueries is empty, the pharmacy sells all kinds of medicines and the name of the pharmacy is returned.

--3.Find the name of pharmacy which sell all kinds of medicine

```

select p1.p_name
From pharmacy as p1
Where NOT EXISTS(
    select distinct(medicine.m_name)
    From medicine
    except
    select distinct(pm.m_name)
    from (pharmacy natural join medicine) as pm
    where pm.p_id = p1.p_id
)

```

$$\Pi_{pharmacy.p_name}(pharmacy) - \Pi_{pharmacy.p_name}(\Pi_{medicine.p_name, medicine.m_name}(medicine) - \Pi_{pharmacy.p_name, medicine.m_name}(medicine \bowtie pharmacy))$$

$$\{t | (\exists p)(p \in pharmacy \wedge p[p_name] = t[p_name] \wedge (\forall m1)(m1 \in medicine \Rightarrow (\exists m2)(m2 \in medicine \wedge m2[m_name] = m1[m_name] \wedge m2[p_id] = p[p_id])))\}$$

Query 3 result:

p_name
character varying (100)

(The empty result table is correct because we didn't have a pharmacy that sells all kinds of medicine in our database.)

Query 4: This query finds the name of the manufacturer which only produces non-prescription medicine. When the user needs a kind of medicine, the user can use the queries to find information about manufacturers and decide which one of them can be a choice. We use the subquery in the where clause to find the name of prescription medicine and the main query finds the manufacturer that does not appear with same tuple as the names output from the subquery.

--4.The name of manufacturer who do not produce prescription medicine

```
Select distinct (manufacturer.f_Name)
From manufacturer natural join medicine
Where medicine.m_name not in
    (Select distinct(medicine.m_name)
    From medicine natural join prescription_medicine
    )
```

$$\begin{aligned}
& \Pi_{manufacturer.f_name} (manufacturer \bowtie_{manufacturer.p_id=medicine.p_id} \\
& \quad (\Pi_{medicine.p_id, medicine.m_name} (medicine) - \\
& \quad \Pi_{medicine.p_id, medicine.m_name} (medicine \bowtie_{prescription_medicine})) \\
& \{t | (\exists f)(f \in manufacturer \wedge f[f_name] = t[f_name] \wedge \\
& \quad (\forall m1)(m1 \in medicine \wedge f[f_id] = m1[f_id] \\
& \quad \Rightarrow \sim \exists(m2)(m2 \in prescription_medicine \wedge \\
& \quad \quad \exists(m3)(m3 \in medicine \wedge m3[m_id] = m2[m_id] \wedge m3[m_name] = m1[m_name]))))\}
\end{aligned}$$

Query 4 result:

	f_name
	character varying (100)
1	TELEVISA CONSUMER PRODUCTS USA
2	Harrison Specialty Co., Inc.
3	Topco Associates LLC
4	PBM Pharmaceuticals Inc.
5	Rebel Distributors Corp

7. Implementation

Our team used PostgreSQL as the database management system, and pgAdmin 4 was used as the development platform to edit and test queries. Because it is not possible to obtain data such as customer SSN and medicine purchase history in real life due privacy issues, the sample data that is currently used in our Pharmacy Database is fake and was randomly generated using Mockaroo. Django, which is Python-based, was used as our web framework to integrate the frontend and database. SQL commands are generated by the default sql setting and the user's input, and executed on the local server. For query 3 and 4 mentioned above in the example queries, we do not have a front end connected to them, and we simply run query 3 and 4 with scripts on pgAdmin 4.

8. Contribution

Sherry:

- Helped to design and create schema
- Wrote the SQL, relational algebra, and tuple relational calculus
- Integrated the frontend and database with Django

- Wrote sql used in web app

Zora:

- Finalized the drawing of the ER diagram
- Helped to generate sample data used in the database
- Improved user interface with HTML and CSS
- Wrote application background and data description

Haomiao Shi:

- Create Relation Schema to database
- Collect, reformat, and Insert Data to Pharmacy Database System
- Set up functional dependencies and Normalization from data
- Determined constraints for functional dependencies and Normalization

9. What We Have Learned

Zora: Completing this project has furthered my knowledge in the implementation of a database. I have now gained a deeper understanding in how databases can be well-applied in real life settings. My skills in employing HTML and CSS have been improved, developing my ability to build a better front end. While making the ER diagram, I have also recognized the importance of the design and utility of the database.

Sherry: In this project, I practiced my skills learned from class like design schema, sql, relational algebra and tuple relational calculus and learned how to use them to build a real life application. I didn't use django to build a web application before. In this project, I learned some basic django framework knowledge and used it to integrate the database and frontend of a web application.

Haomiao Shi: From this project, I learned how to insert data to the database by using the postgresSQL in pgAdmin4. I used to insert data into the database by using SQL in datagrip. Moreover, I learned the way to classify different functional dependencies and the way to add constraints to these functional dependencies.