

Vaccine History and Stock Database **(Clinical Information System)**

Business goal and company objective

A national vaccine institute would like to build a central database for estimating herd immunity and distributing vaccines to specific areas with insufficient vaccines. This coordination requires large data gathering among healthcare centers and commercial vaccine data to share important information and overcome these problems.

The comprehensive repository of vaccination records is an important tool in modern healthcare, enabling healthcare providers to access critical information about patient's immunization history quickly and accurately. In the case of outbreaks or pandemics, a central vaccine database allows the government to prepare the information for rapid response and management by calculating the vaccinated population and susceptible host. Following this, they would utilize this information with reported cases to assess the pandemic, implement the preventive policy and distribute the proper number of vaccines to the high-cautious area.

Furthermore, the adverse effects can be tracked to monitor the vaccine safety and efficacy which is beneficial for recalling after finding the and research purpose. Moreover, when the patients immigrate to other states, the new healthcare provider can easily track to the previous vaccine history for continuing the vaccine program or considering as a supporting document for differential diagnosis.

Therefore, the database should store, connect, and organize patient and healthcare provider data, including vaccine products and stocks. The vaccine order will be the primary step for tracking the overall operation between healthcare and patients.

User Requirements

A patient taking the service from healthcare providers will receive the unique Patient ID (PID). Other general information is name (first name and last name), gender, date of birth, address (street, city, state, zip code), preferred language, phone number and race.

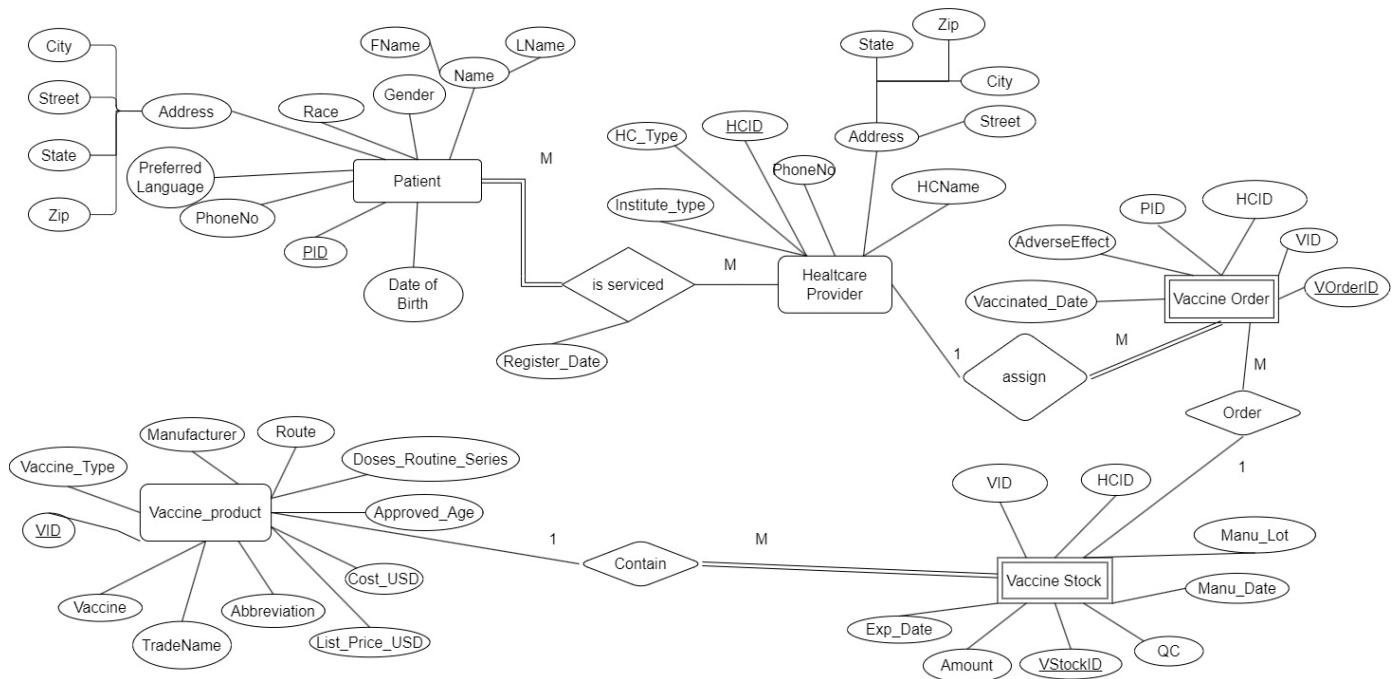
Meanwhile, patients can register to many healthcare centers, but they have only one PID for their whole life. Each patient is obligated to register at least one healthcare center.

Healthcare providers also have their own code (HCID) and the institutes data: healthcare type (hospital, clinic, drug store) healthcare name, institute type (public, private, academic), address (street, city, state, zip code) and phone number.

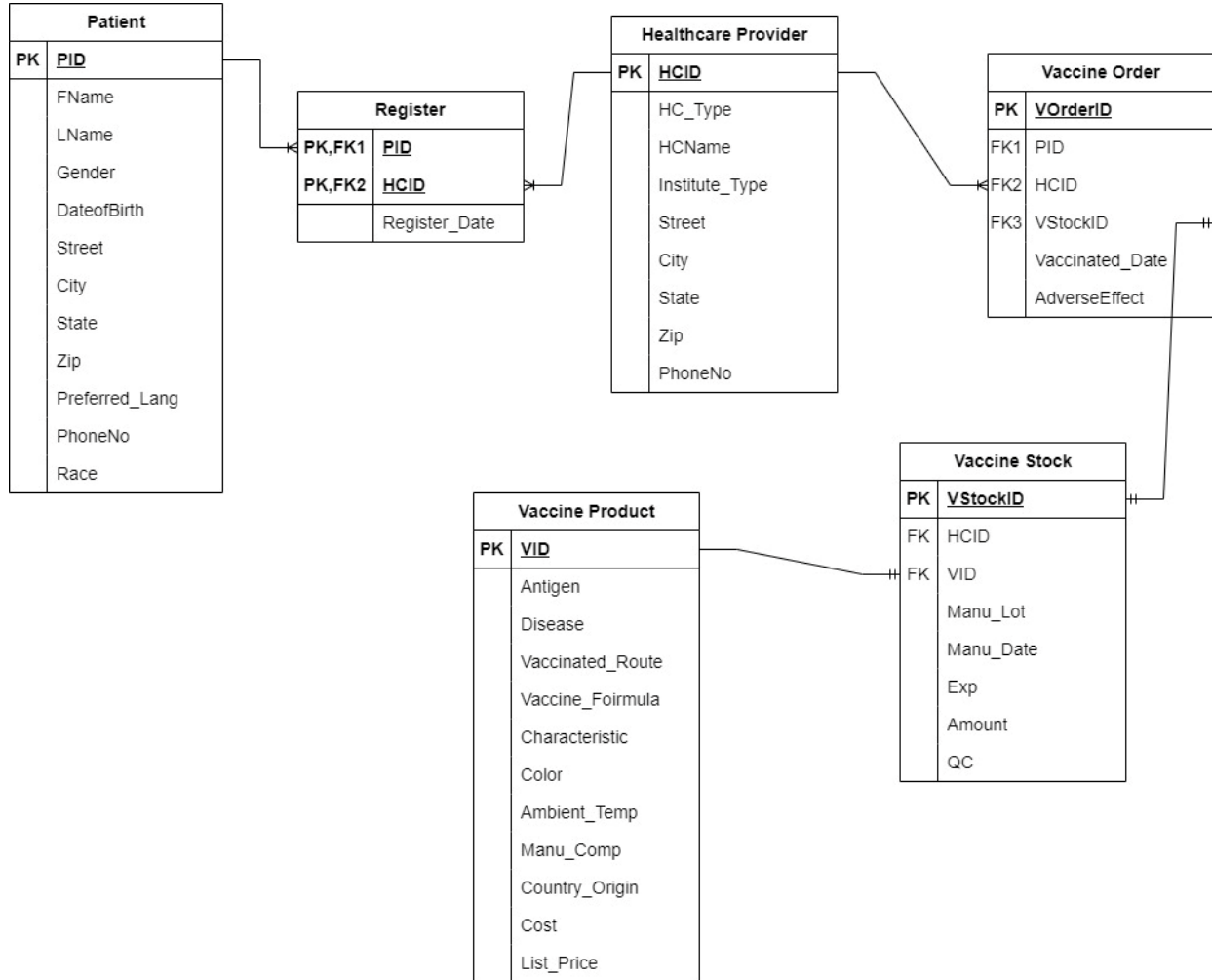
For the vaccine data, beginning with the vaccine product which has the unique vaccine ID (VID) and general information, namely vaccine name, trade name, abbreviation, manufacturer, injected route, the number of dose routine series, approved age, vaccine type, cost and list price. After the healthcare center receive the vaccine products from the distributor, they will record the manufactured lot number, manufactured date, expired date, amount, and quality checking for safety. Meanwhile, the same vaccine ID can be manufactured repeatedly but the lot numbers are changed due to the manufactured date and expired date. Each lot also has unique ID.

When patients come to a healthcare provider to request vaccination, the healthcare center will give a vaccine order (VOrderID) and stock ID to track the assigned vaccine. Simultaneously, the healthcare center will record the vaccination date and adverse effects.

Conceptual Data Modeling



Logical Database Design



Implementation

```
CREATE TABLE Healthcare_Provider
```

```
( HCID number(6),
```

```
HCName VARCHAR2(50) not null,
```

```
HC_Type VARCHAR2(50) not null,
```

```
Institute_Type VARCHAR2(50),
```

```
Street          varchar2(50),
```

```
City            varchar2(50),
```

```
State           varchar2(50),
```

```
Zip             number(5),
```

```
PhoneNo         varchar2 (25),
```

```
-- healthcare information
```

```
-- Healthcare center ID
```

```
-- To specify the name of healthcare center
```

```
-- Health center, Clinic, Drug store, Hospital
```

```
-- Private, Public, Academic
```

```
-- Address information (Street, City, State, Zip)
```

```
-- Phone number
```


Jittrapol Sutejitsiri u1538222

AdverseEffect VARCHAR2(225), -- The detail of adverse effects contain the long description
constraint Order_PK primary key (VOrderID), -- Assigned VOrderID to primary key
constraint PID_FK foreign key (PID) references Patient, -- refer PID to Patient table
constraint HCID_FK foreign key (HCID) references Healthcare_Provider, -- refer HCID to healthcare table
constraint VStockID_FK foreign key (VStockID) references Vaccine_Stock); -- refer VStockID to vaccine stock

CREATE TABLE Vaccine_Product -- Give vaccine information
(VID NUMBER(6), -- Vaccine ID to identify each vaccine formula
Vaccine VARCHAR2(50), -- name of disease that vaccine can arouse the immunity
TradeName VARCHAR2(50), -- Commercial vaccine name in the market
Abbreviation VARCHAR2(50), -- Brief name of this vaccine/ company
Manufacturer VARCHAR2(50), -- Manufacturing company
Route VARCHAR2(50), -- Vaccinated route to patient
Doses_Routine_Series VARCHAR2(50), -- The number of reimmunize to reach a protective immunity
Approved_Age VARCHAR2(50), -- Approved age for vaccination in this product
Vaccine_Type VARCHAR2(10), -- Inactivated, live-attenuated, mRNA type
Cost_USD NUMBER (6,2), -- Cost of vaccine in USD currency
List_Price_USD NUMBER (6,2), -- Recommend list price of vaccine in USD currency
primary key (VID));

CREATE TABLE Vaccine_Stock -- Vaccine stock table contains detail for storing product
(VStockID NUMBER (6), -- Vaccine stock ID
HCID NUMBER (6), -- Healthcare ID
VID NUMBER (6), -- Vaccine ID
Manu_Lot VARCHAR2(10), -- Manufacturing lot number contain manufacturing date/expired date
Manu_Date DATE, -- Manufacturing date
Exp_Date DATE, -- Expired date
Amount NUMBER(10), -- The number of stock vaccine in this lot
QC NUMBER(1), -- Quality checking (1 – pass, 0 -not pass)
primary key (VStockID), -- Assigned VStockID as a primary key
constraint HCID_FK foreign key (HCID) references Healthcare_Provider, -- Refer HCID to healthcare table
where is possession to the vaccine

constraint VID_FK foreign key (VID) references Vaccine_Product); -- Refer VID to vaccine product table

```
CREATE VIEW P_Age as -- create view for calculating patient age
SELECT PID, DateofBirth, -- select PID and date of birth from patient
      (strftime('%Y', 'now') - strftime('%Y', DateofBirth)) - table and change type of date to separate
      (strftime('%m-%d', 'now') < strftime('%m-%d', DateofBirth)) AS age sting for easy to calculate
from Patient;
```

Populate sample data

I randomized the dataset from Python using faker library to simulate general information (name, address, phone no.) and birth date of patients. Some random datasets that I wanted to focus, were derived from my own design such as race, preferred language, healthcare type. After that, I manually transferred the data from Python to SQLite. For the primary key ID of each table, I ran the sequence of numbers and concatenated them with the random data to insert into the table. However, the vaccine product data was brought from CDC vaccine recommendation in adults, and the influenza, dengue, and pneumococcal vaccines were chosen, but the cost and list price were randomized from the range of the actual price.

All of the data is randomized except the vaccination date, which was randomized and range between the manufactured date and the expired date, and the expired date, which was counted 3 years after the manufactured date. Therefore, some data might be conflicted with other fields. For example, some patients might have a registered date before they were born, or their age might not match the approved age.

At this point, I assigned only one healthcare center to one patient to prevent duplication of vaccine orders after randomizing the vaccine order to Stock ID.

To generate vaccine orders, healthcare centers were selected and then copied the PID who registered in the selected healthcare for the designated list. After that, the data was randomized with VstockID and vaccinated date.

```

from faker import Faker
import datetime
import random

# Create a Faker instance
fake = Faker()

# Generate a random date between two dates
start_date = datetime.date(1999, 1, 1)
end_date = datetime.date(2024, 12, 1)
random_date_in_range = fake.date_between(start_date=start_date, end_date=end_date)
next_two_y = random_date_in_range + datetime.timedelta(days= 3*365)
HCID = [200001,200002,200003,200004,200005,200006,200007,200008,200009,200010]
VID = [300001,300002,300003,300004,300005,300006,300007]

for i in range(25):
    VstockID = 400001
    VstockID += i
    random_date_in_range = fake.date_between(start_date=start_date, end_date=end_date)
    next_two_y = random_date_in_range + datetime.timedelta(days= 3*365)
    print(f"({VstockID},{random.choice(HCID)},{random.choice(VID)},{random_date_in_range},{next_two_y},{random.randint(1000,10000)},'1')")

```

✓ 0.0s

```

(400001,200006,300001,'2020-05-13','2023-05-13','1930','1')
(400002,200006,300006,'2022-08-06','2025-08-05','3246','1')
(400003,200010,300005,'2011-09-14','2014-09-13','5570','1')
(400004,200010,300007,'2009-05-10','2012-05-09','5982','1')
(400005,200005,300005,'2015-05-27','2018-05-26','8597','1')
(400006,200004,300003,'2010-02-05','2013-02-04','6568','1')

```

Frequently used queries for data extraction and clinical report

- The number of patients who is vaccinated between 2019 and 2022 grouped by vaccine name
 - In order to find the number of vaccine doses in each year, it is important for conclude the report in each year.
 - First, I selected year format (transform to string), vaccine name and count of patient number. But Vaccine_Order table did not have the vaccine name, so I had to join with the Vaccine_Product table and assigned the year between 2019 and 2023. Following this, the data was grouped by vaccine name and ordered by year.

```

1 SELECT strftime('%Y', Vaccine_Order.Vaccinate_Date) AS Year
   ,Vaccine_Product.Vaccine, COUNT(Vaccine_Order.PID) AS
   Vaccinated_cases FROM Vaccine_Order
   . JOIN Vaccine_Product ON Vaccine_Order.VID = Vaccine_Product
   .VID
   . WHERE Vaccinate_Date BETWEEN '2019-01-01' AND '2023-12-31'
   . GROUP BY Vaccine_Product.Vaccine, Year
5 ORDER BY Year;

```

Execute SQL Stop Query *<Untitled

Year	Vaccine	Vaccinated_cases
(empty)	(empty)	(empty)
2019	Influenza	22
2019	Pneumococcal	7
2020	Influenza	38
2021	Influenza	31
2022	Influenza	8

2. The average age and number of patients who was vaccinated pneumococcal

- Due to the susceptible age, old age people should be vaccinated. Therefore, healthcare center can see the average age for focusing on the suitable target group.
- I assigned 'with clause' to people who are vaccinated pneumococcal vaccine because it was rather complicated, and I had to join the table specified 'Pneumococcal vaccine'. For age calculation (figure 2.1,2.2), I created view for age, derived from minus the birth date and current date

```

1 CREATE VIEW P_Age as
. SELECT PID, DateofBirth,
.      (strftime('%Y', 'now') - strftime('%Y', DateofBirth)) -
.      (strftime('%m-%d', 'now') < strftime('%m-%d', DateofBirth)) AS age
. from Patient;
.

```

Figure 2.1 Create view for age calculation

PID	DateofBirth	age
(empty)	(empty)	(empty)
▶ 100001	1989-07-31	35
100002	1935-11-12	89
100003	1938-08-08	86
100004	1984-02-26	40
100005	1937-03-29	87

Figure 2.2 View table to store age data

```

. WITH Vac_Pneu AS (
10  SELECT Vaccine_Order.PID, Vaccine_Product.Vaccine
.    FROM Vaccine_Order
.   JOIN Vaccine_Product ON Vaccine_Order.VID = Vaccine_Product.VID
.  WHERE Vaccine_Product.Vaccine LIKE 'Pneumococcal'
. )
. SELECT AVG(P_Age.age) AS average_age, count(P_Age.PID) AS number
. FROM P_Age
17 JOIN Vac_Pneu ON P_Age.PID = Vac_Pneu.PID;

```

◀ ◁ ▷ ▶ + - ↑ ✓ ✕ ↺
 Execute SQL Stop Query

average_age	number
(empty)	(empty)
▶ 45.7215189873418	79

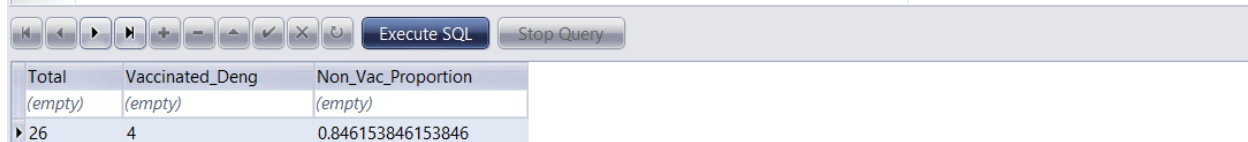
Figure 2.3 Result of complete queries for average age and number of pneumococcal vaccinated

3. Count the number of people who is not vaccinated dengue vaccine in Texas, in 2012
 - To estimate the herd immunity, the healthcare centers must find the percentage of vaccinated and non-vaccinated people.
 - I created the 'Total_Pop' from the Patient table in Texas and 'Vaccinated number' in Dengue, in 2012. After that, I calculated the proportion by multiplying 1.0 for change the integer to float type.

```

. --the proportion of people who is not vaccinated Dengue vaccine in Texas, in 2021
20 -- Find the total patient
. WITH Total_Pop AS (
.   SELECT COUNT(PID) as Total FROM Patient
.   WHERE State = 'Texas'
. ),
. -- Find the number of vaccinated Dengue people
. VaccinatedNo AS (
.   SELECT COUNT(DISTINCT Vaccine_Order.PID) AS Vaccinated_Deng FROM Vaccine_Order
.   JOIN Vaccine_Product ON Vaccine_Order.VID = Vaccine_Product.VID
.   WHERE Vaccine_Product.Vaccine LIKE 'Dengue' AND Vaccinate_Date BETWEEN '2012-01-01' AND '2012-12-31'
30 )
. -- Find the proportion
. SELECT Total, Vaccinated_Deng, 1-(Vaccinated_Deng*1.0/Total) as Non_Vac_Proportion From Total_Pop, VaccinatedNo

```



Total	Vaccinated_Deng	Non_Vac_Proportion
(empty)	(empty)	(empty)
26	4	0.846153846153846

Challenges, lessons learned and rooms for improvement

During the project implementation, I have learned about the numerous aspects of using SQL skills in real life. First of all, I have to combine the lessons from every single week, such as logical and physical relational databases, and rearrange my understanding to create the project from scratch. This means that I should understand the lesson and find the further or more complex things to apply to the project. Another point is that applying Python to SQL is absolutely helpful. I used to think that each computer language was separate and might not associate with each other because I did not have a background in computer science or programming. After I learned both languages and found some other resources, I tried to apply them and also found that there are numerous things to help work easier, especially in a random samples. Therefore, I will find the method to link SQL to Python in order to automatically insert the sample or update the data. Finally, programming is superior to logical thinking. This project encouraged me to break complex tasks into small sections. For example, if I am supposed to create a complex query (the number of people who are not vaccinated with the influenza vaccine in Texas in 2021), I have to pull the information from the Texas people, influenza vaccinated people, and those vaccinated in 2021. This thinking process will help me relieve when I suffer from complex tasks and can be applied to real life. The next step to improve my database skills is to iterate the adverse effect from the extended

phrase to a small chunk because I would like to classify the severity and count the symptom's frequencies. Some attributes may contain a long phrase as a general input, leading to unidentified and invalid data analysis. This skill will leverage me in a variety of database management.