GROUP 21

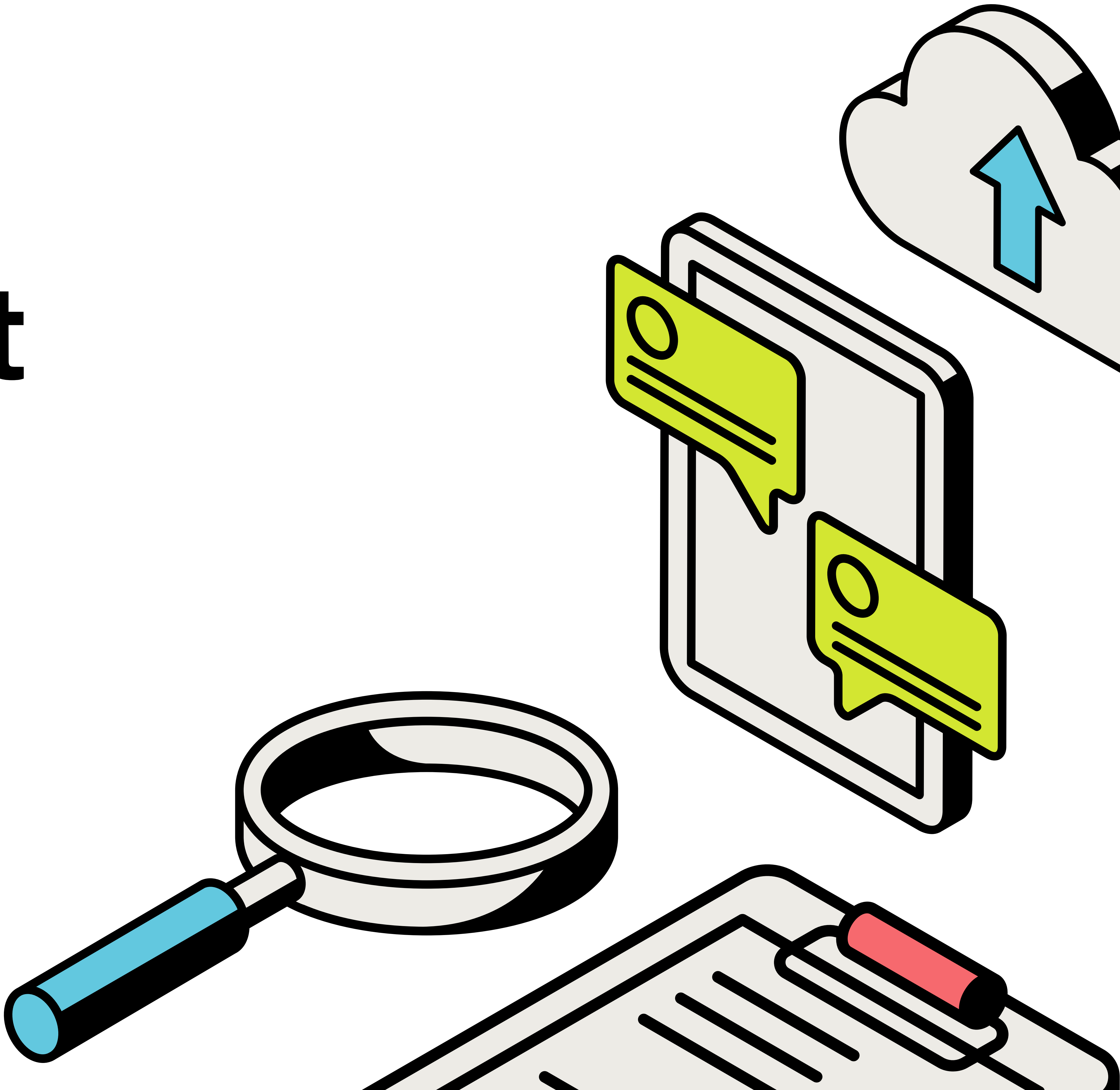
Group Project Presentation

To Minh Duy

Nguyen The Hung

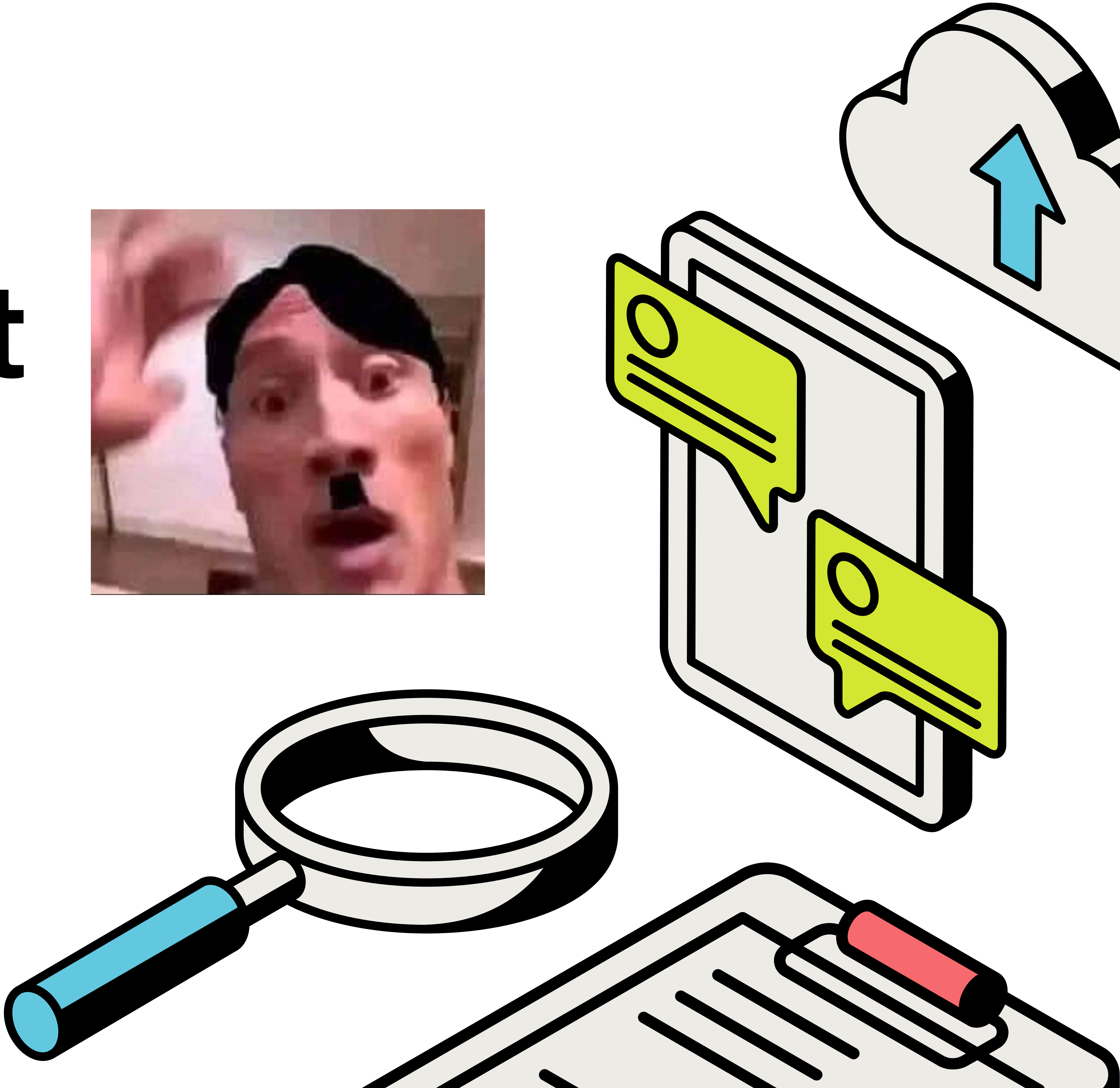
Cao Thuy Duong

Tran Minh Quang



GROUP 21

Group Project Presentation



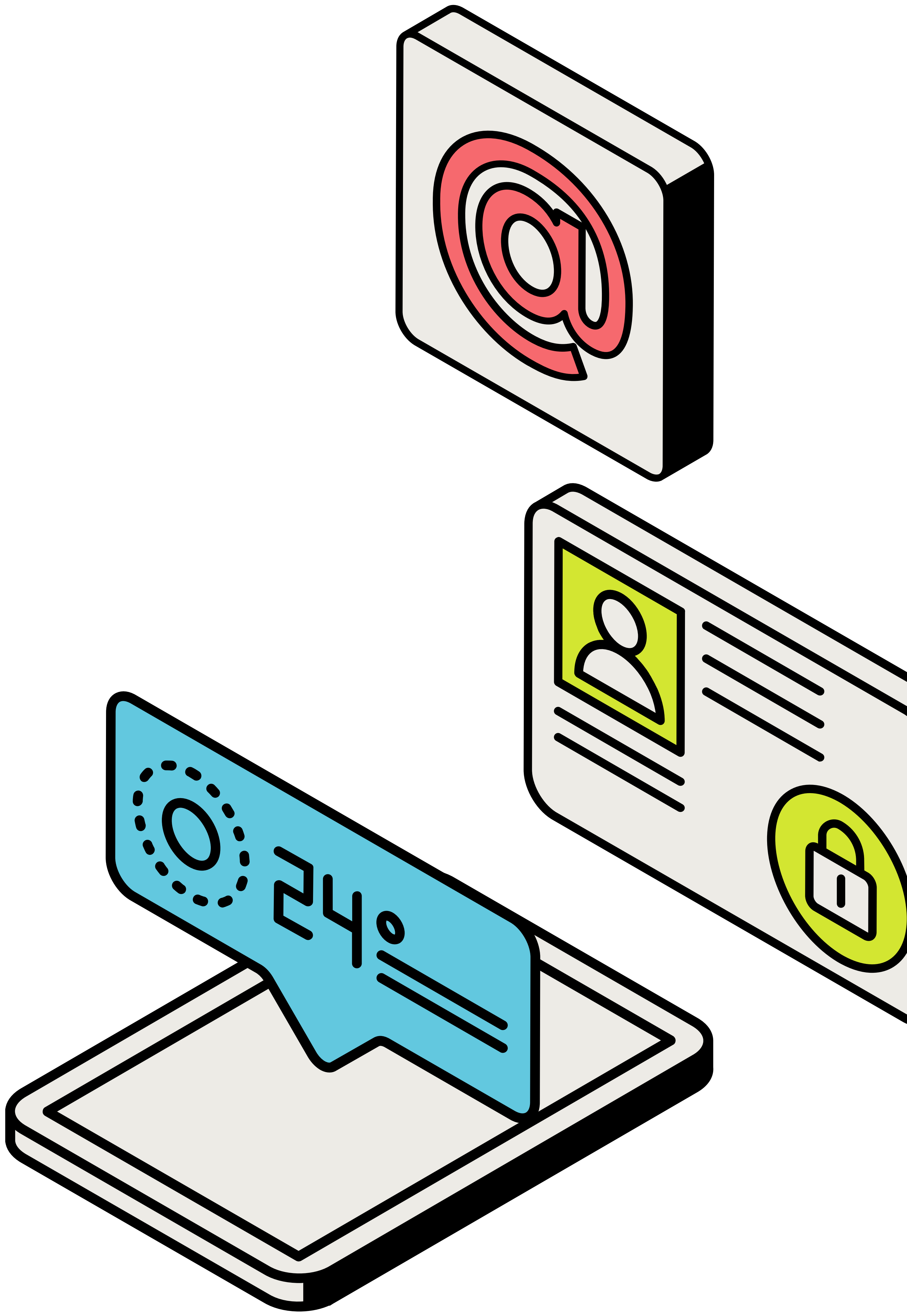
To Minh Duy

Nguyen The Hung

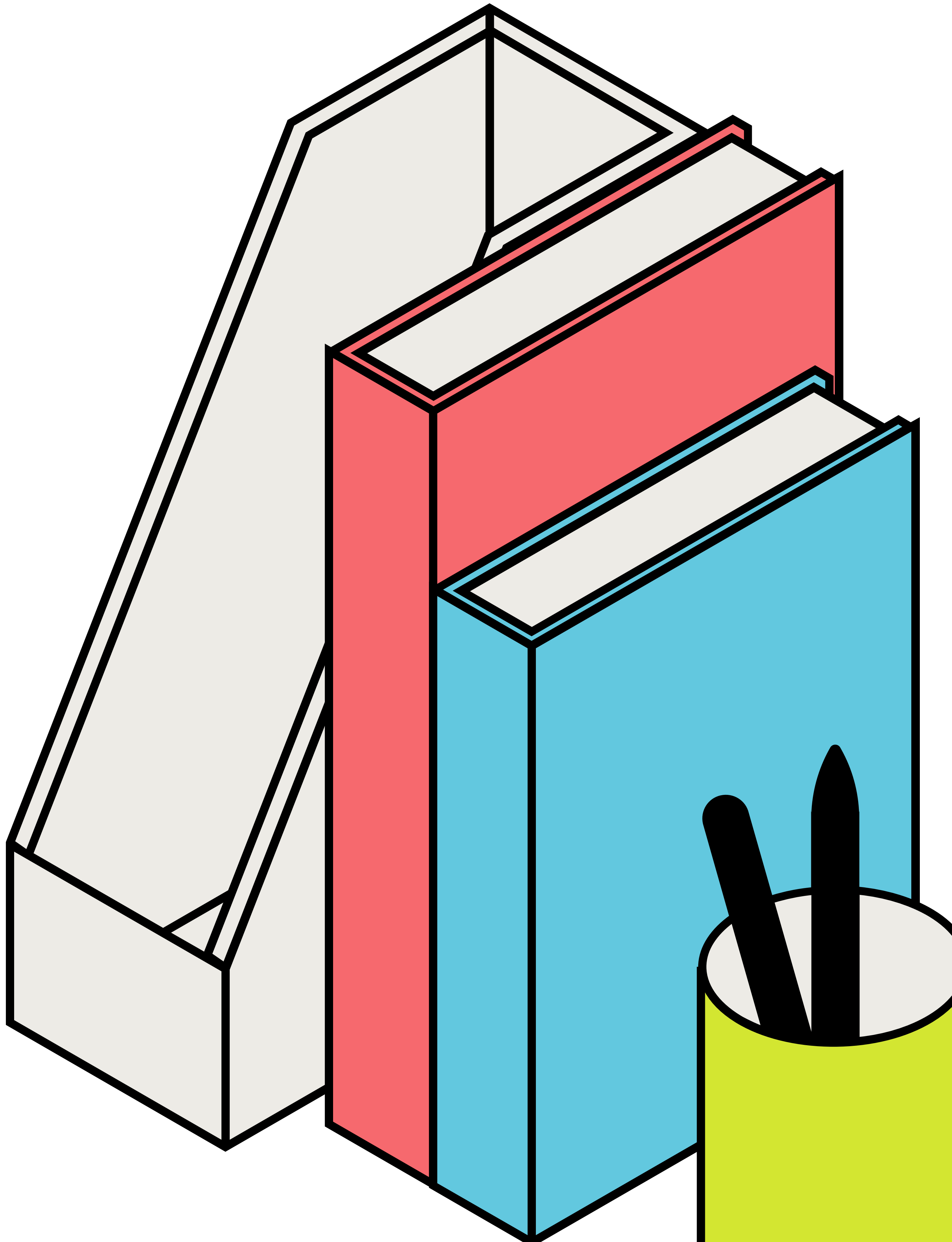
Cao Thuy Duong

Tran Minh Quang

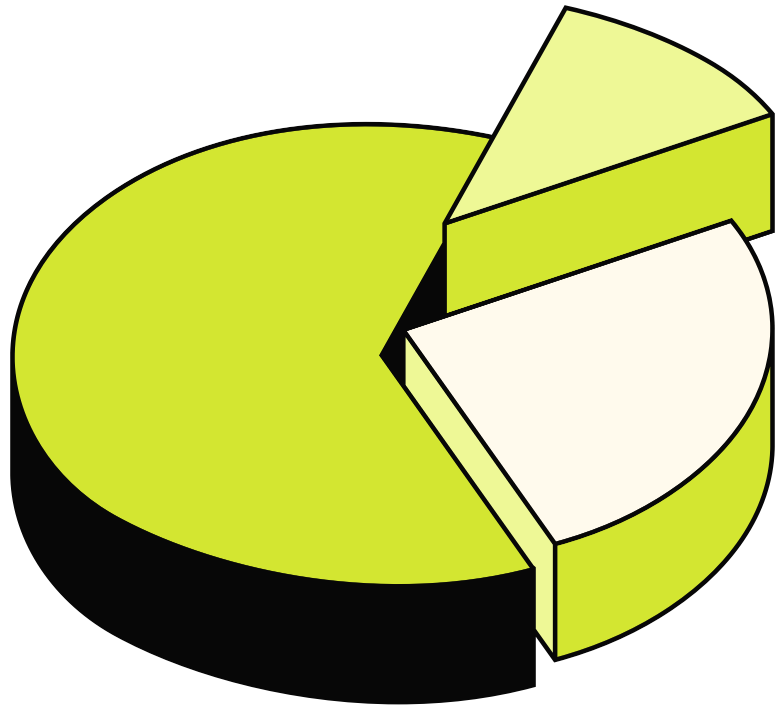
Introduction



In this document:



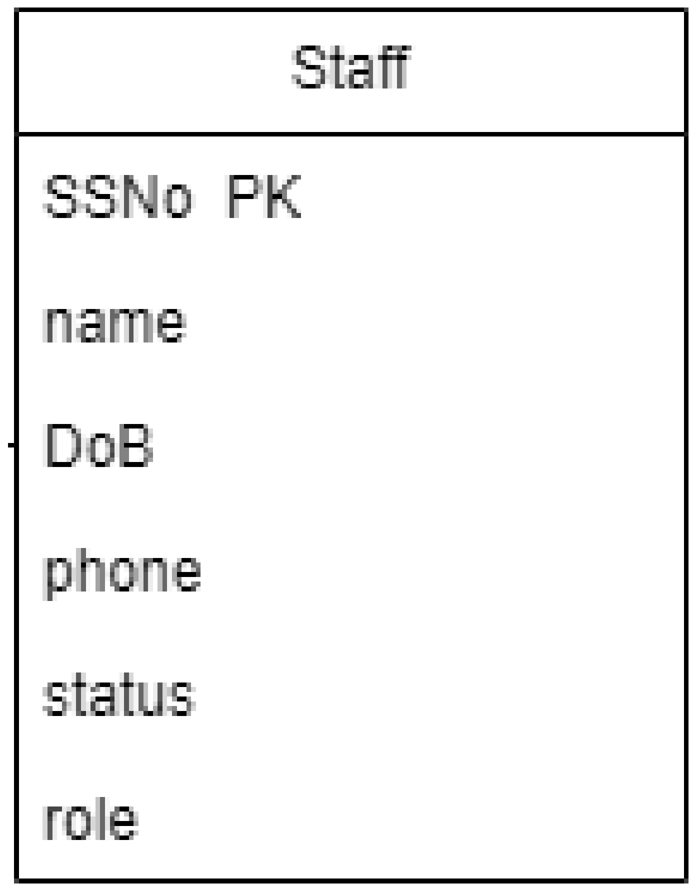
UML and
Relational Schema



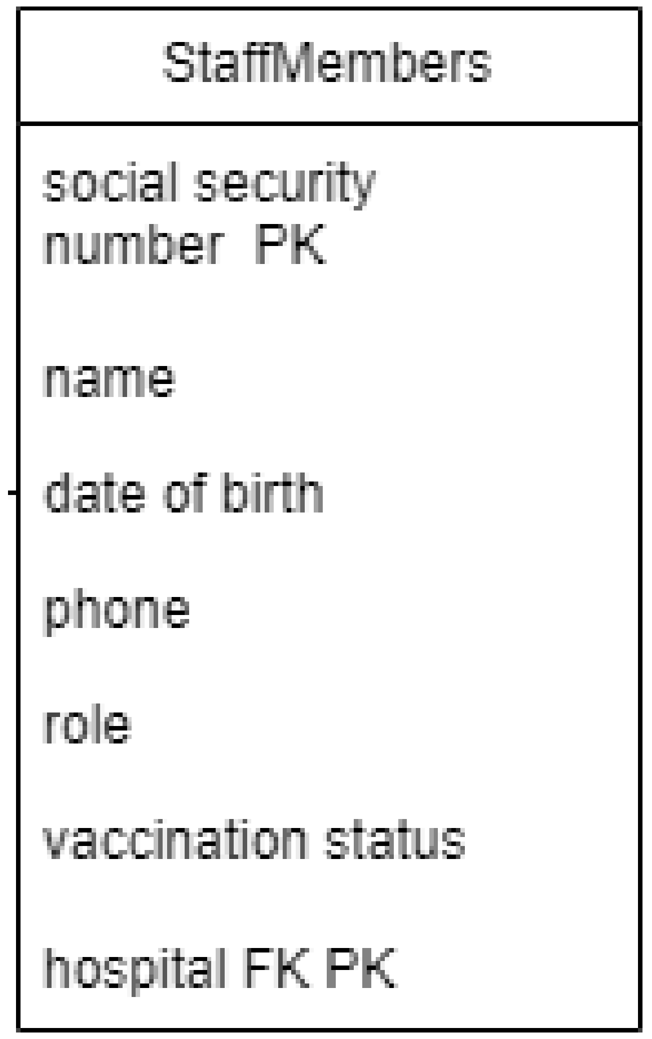
UML

Classes and attributes are renamed.
New classes and attributes are added, according to the database
Foreign key was defined

e.g. Class Staff -> StaffMembers.
Its attributes SsNo, DoB, status -> "social security number", name, "date of birth" "vaccination status"
Attribute "hospital" has been added and defined as foreign key



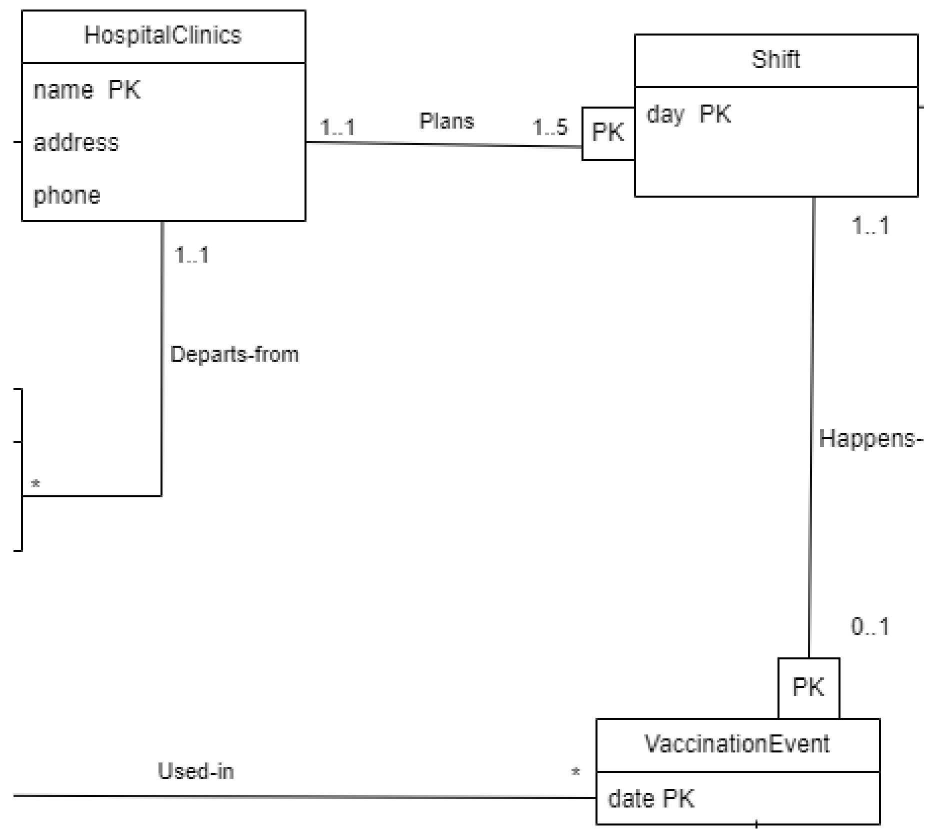
Former



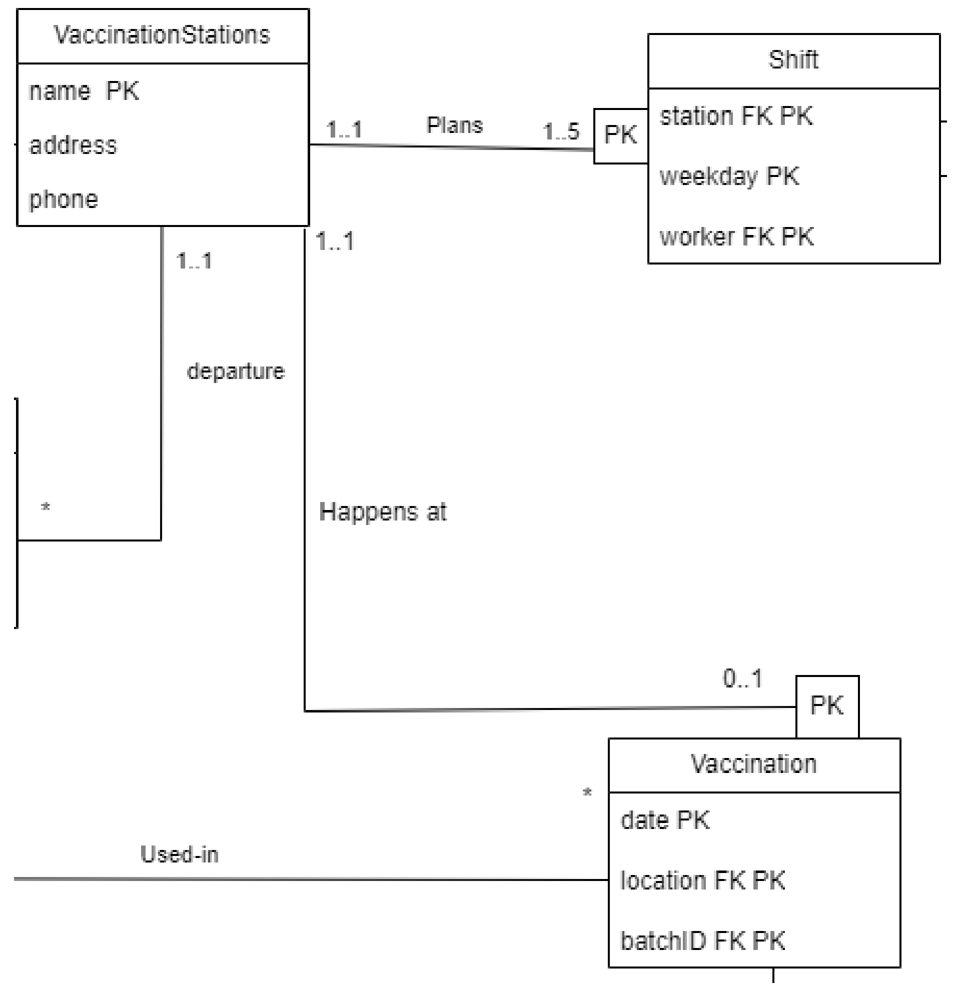
Latter

UML

Class 'Vaccination' is now directly associated to class 'VaccinationStations'



Former



Latter

Figure 1.1: The former version of our UML model

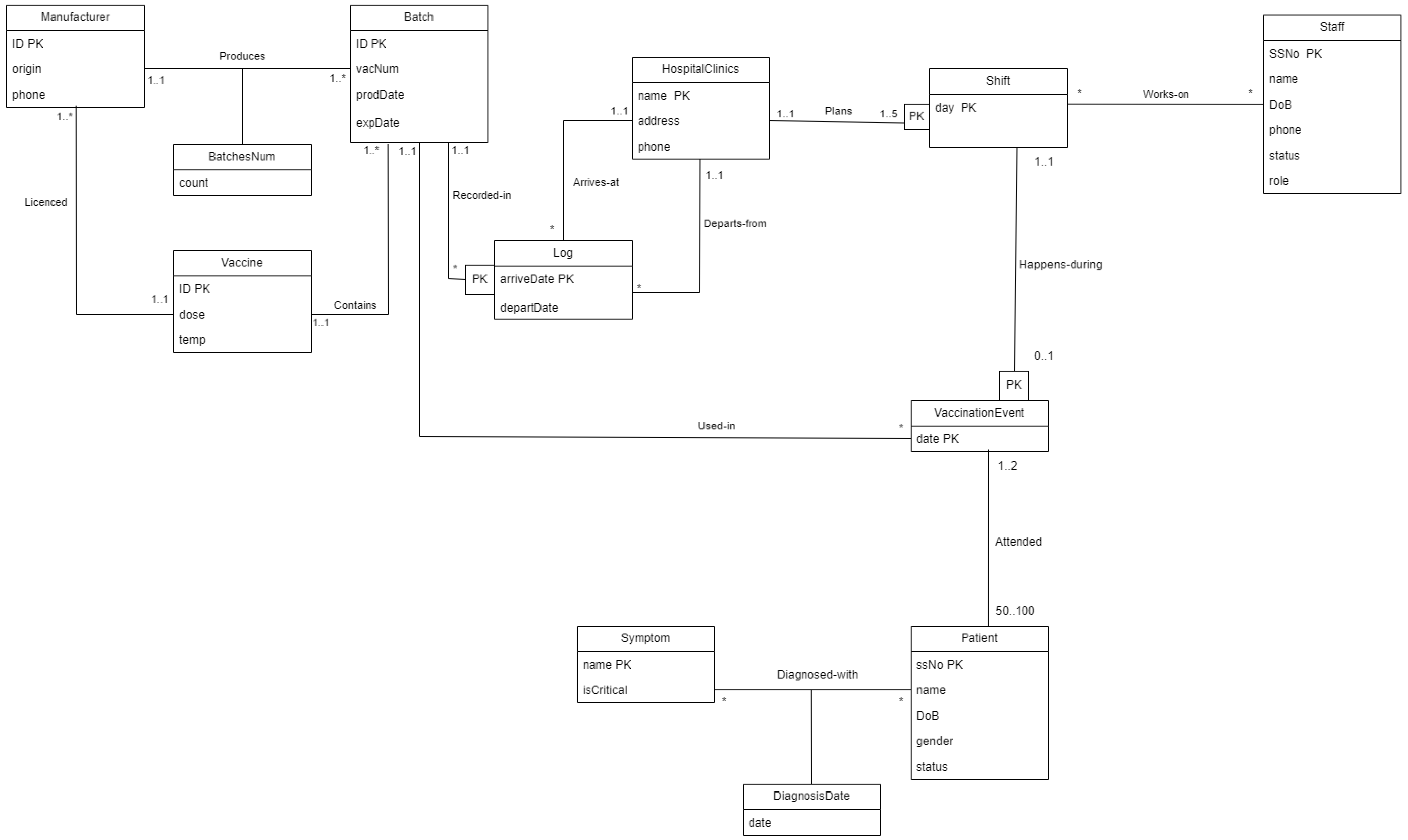
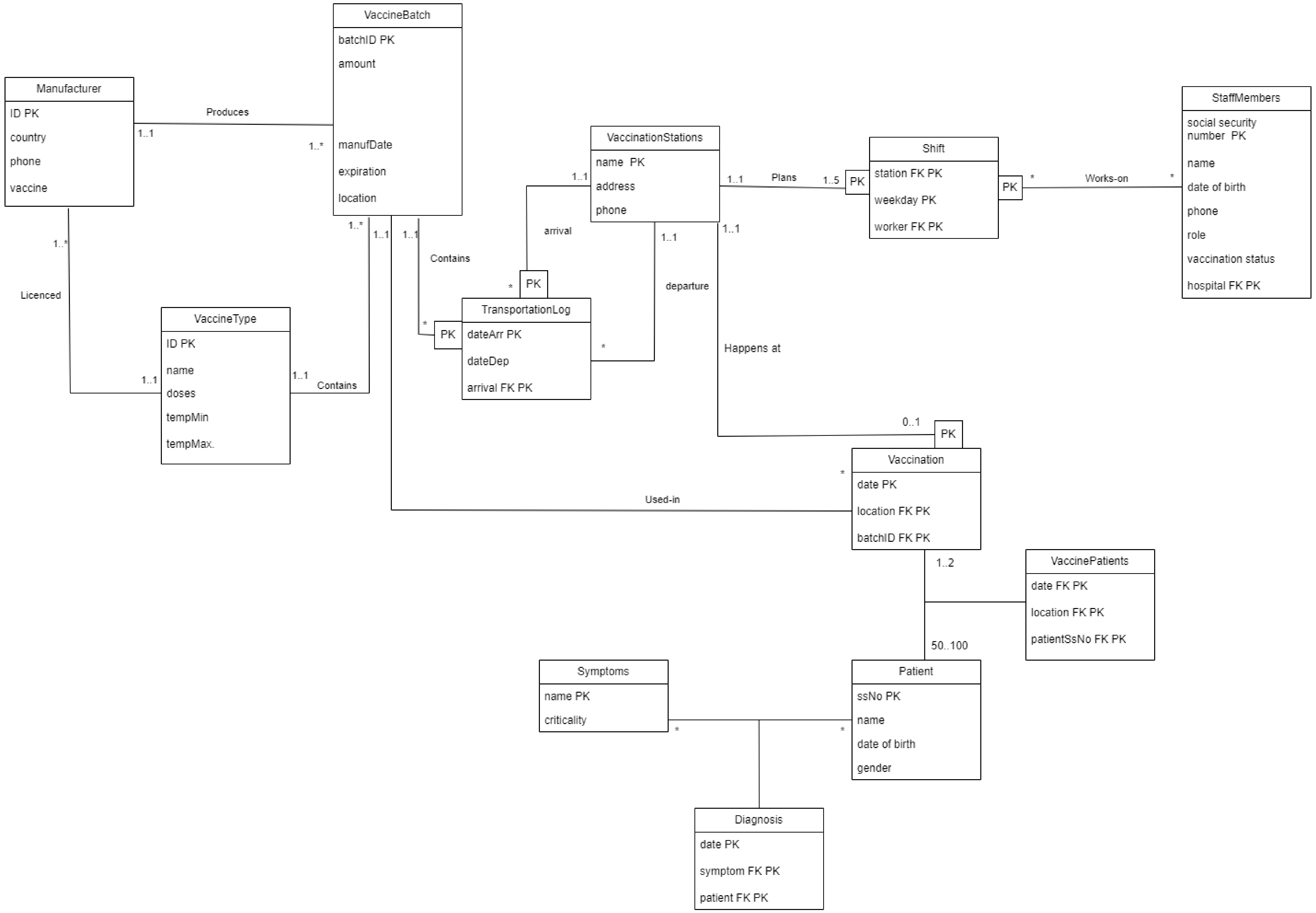


Figure 1.2: The updated version of our UML model



The former vs the updated version of our relational schema

Former	Latter
Manufacturer(ID, origin, phone, vaccine)	Manufacturer(ID, country, phone, vaccine)
Batch(ID, vacNum, prodDate, expDate, vaccineType, manuID)	VaccineBatch(batchID, amount, type, manufacturer, manufDate, expiration, location)
Vaccine(ID, dose, temp)	VaccineType(ID, name, doses, tempMin, tempMax)
HospitalClinics(name, address, phone)	VaccinationStations(name, address, phone)
Log(arriveDate, departDate, batchID, arrival, departure)	TransportationLog(batchID, arrival, departure, dateArr, dateDep)
Plans(hospitalClinics, shift)	
Shift(hospitalClinics, day)	Shift(station, weekday, worker)
Staff(SSNo, name, DoB, phone, status, role)	StaffMembers(“social security number”, name, “date of birth”, phone, role, “vaccination status”, hospital)
WorksOn(shift, staff)	
VaccinationEvent(shift, date, hospitalClinics)	Vaccination(date, location, batchID)
HappensDuring(event, shift)	
Patient(ssNo, name, DoB, gender, status)	Patient(ssNo, name, “date of birth”, gender)
Attended(hospitalClinics, shift, date, patient)	VaccinePatients(date, location, patientSsNo)
Symptom(name, isCritical)	Symptoms(name, criticality)
DiagnosedWith(symptom, patient, diagDate)	Diagnosis(patient, symptom, date)

Relational Schema

1	There were some renaming, removal and addition of relations and attributes
2	Association relations ‘Plans’, ‘WorksOn’, ‘HappensDuring’ have been eliminated
3	Relation ‘VaccineBatch’ has an additional attribute ‘location’, which indicates the current location of the batch.
4	Attribute ‘temp’ of relation ‘Vaccinetype’ is changed into ‘tempMin’ and ‘tempMax’.
5	Changes in the primary key of relation ‘TransportationLog’
6	Relation ‘Shift’ has an additional attribute ‘worker’, and all its attributes are primary key
7	Relation ‘StaffMember’ has additional attributes ‘vaccination status’ and ‘hospital’
8	Relation ‘Vaccination’ (previously known as ‘VaccinationEvent’) has relation ‘batchID’ instead of ‘shift’
9	Attribute ‘status’ is eleminated from relation ‘Patient’
10	New relation ‘VaccinePatients’ is created

Note: the full SQL code will be attached at the end of this presentation

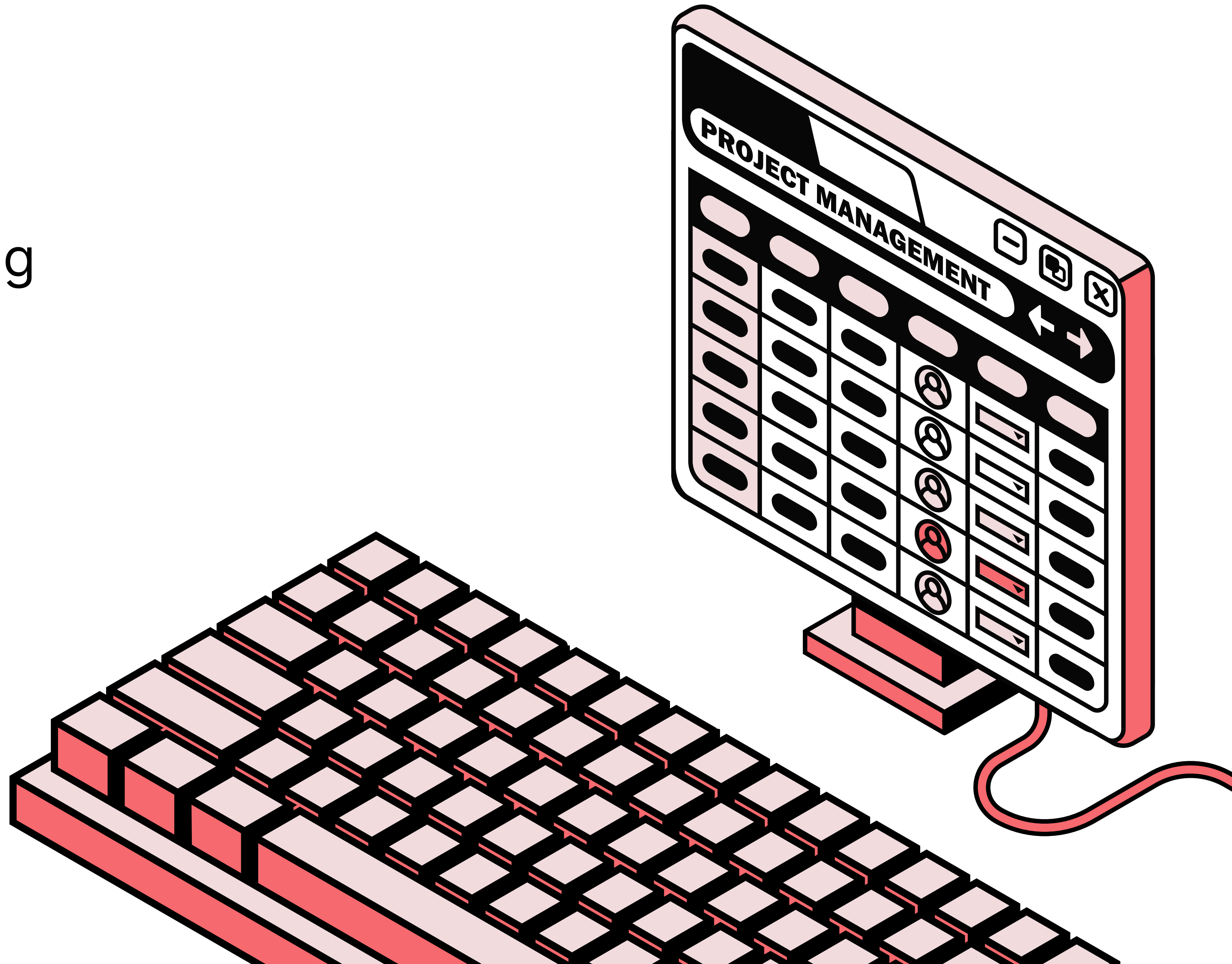
SQL Tables and Queries

Done with the UML model and the relational schema, we proceeded to create SQL tables and queries using DBeaver for later usage during the Python implementation process.

Tables

Create required tables with suitable attributes, taking into consideration data types and conditions.

Set primary keys and foreign keys.



Note: the full SQL code will be attached at the end of this presentation

SQL Tables and Queries

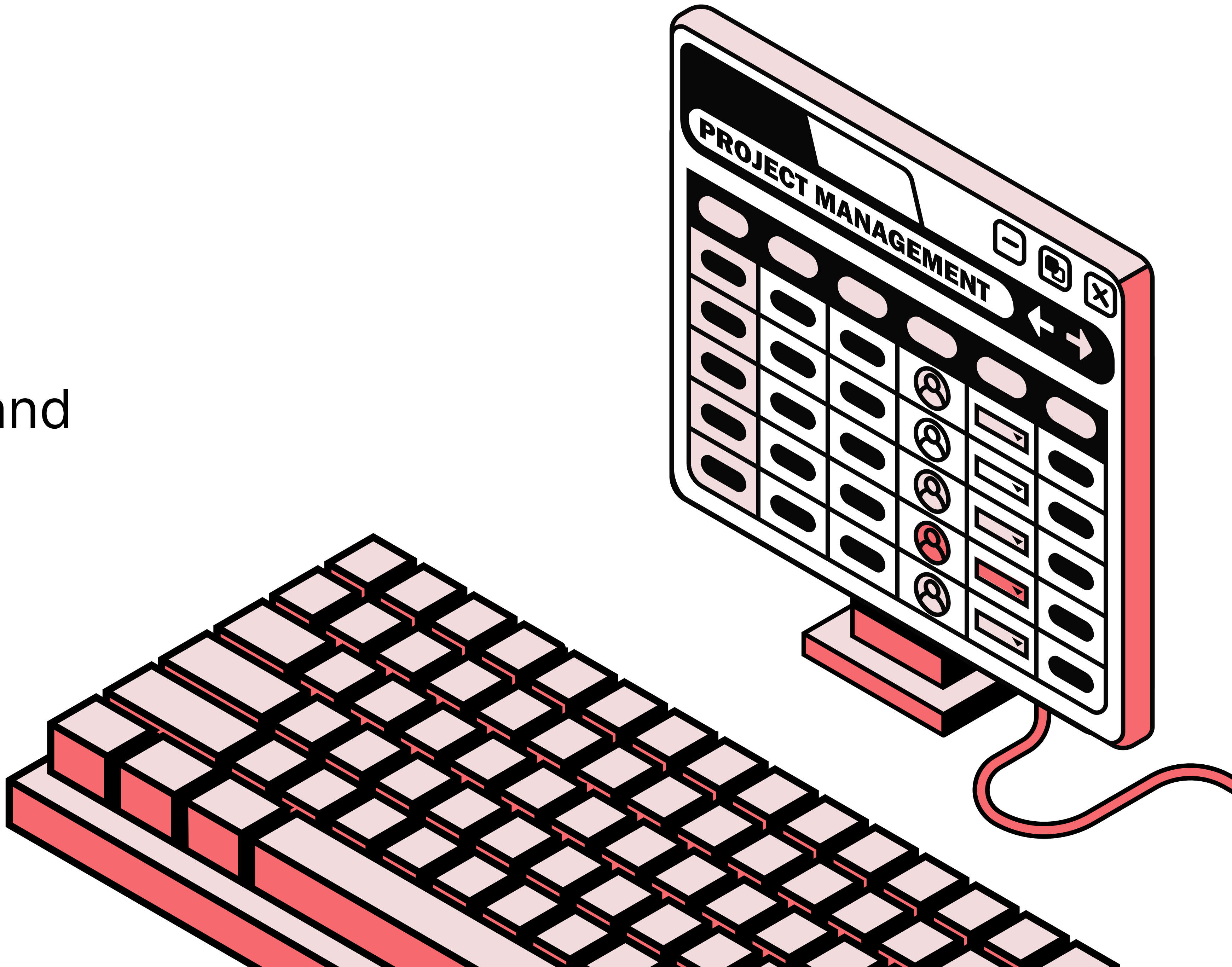
Done with the UML model and the relational schema, we proceeded to create SQL tables and queries using DBeaver for later usage during the Python implementation process.

Queries

Queries were written based on the requirements.

Select necessary attributes from relevant relations and set some restrictions to limit output (if needed).

Use some aggregate functions and subqueries.



Python Implementation

We used psycopg2, sqlalchemy, and pandas to create the database with the given credential, and populate that database

Create tables

1. Create an SQL file using DBeaver and a local test database to make testing easier .

2. Use function run_sql_from_file() to create the tables using that sql file.

3. Use DBeaver to connect to the database to visually check if the tables had been successfully created.

Populate the database

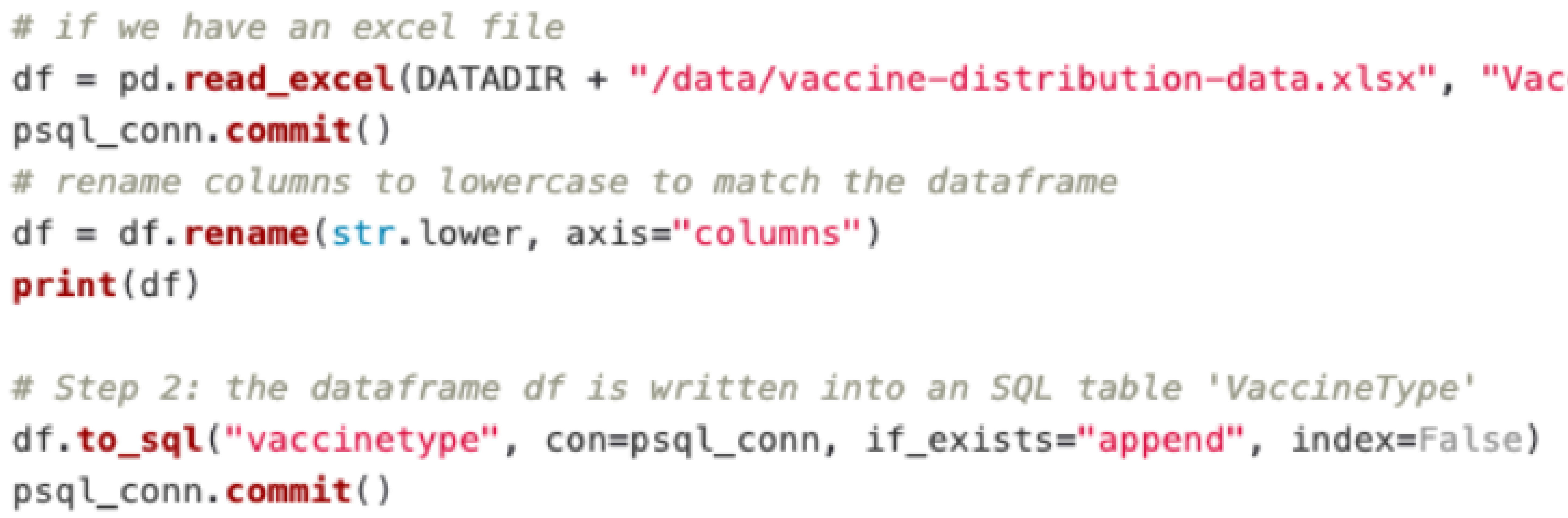
1. Use read_excel() from Pandas library to read data from one sheet at a time to a dataframe,

2. Use to_sql() to import that dataframe to the database.

3. Repeat this process for every sheet in the data file.

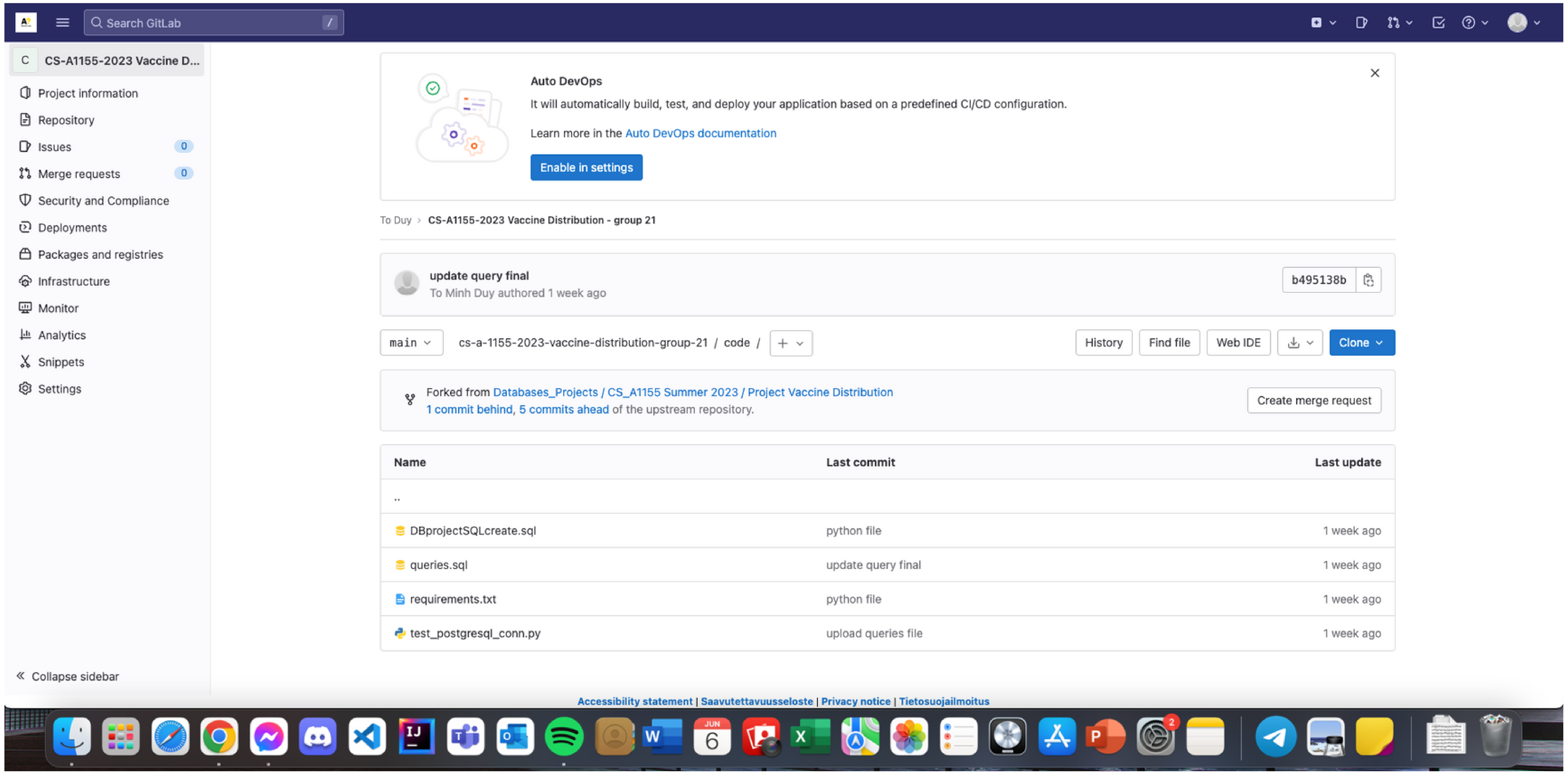
Python Implementation

We used psycopg2, sqlalchemy, and pandas to create the database with the given credential, and populate that database



```
# if we have an excel file
df = pd.read_excel(DATADIR + "/data/vaccine-distribution-data.xlsx", "VaccineType")
psql_conn.commit()
# rename columns to lowercase to match the dataframe
df = df.rename(str.lower, axis="columns")
print(df)

# Step 2: the dataframe df is written into an SQL table 'VaccineType'
df.to_sql("vaccinetype", con=psql_conn, if_exists="append", index=False)
psql_conn.commit()
```

Gitlab cooperation



CS-A1155-2023 Vaccine D...

Project information
Repository
Issues 0
Merge requests 0
Security and Compliance
Deployments
Packages and registries
Infrastructure
Monitor
Analytics
Snippets
Settings

Collapse sidebar

Auto DevOps

It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.

Learn more in the Auto DevOps documentation

Enable in settings

To Duy › CS-A1155-2023 Vaccine Distribution - group 21

update query final
To Minh Duy authored 1 week ago
b495138b

main cs-a-1155-2023-vaccine-distribution-group-21 / code /

History Find file Web IDE Clone

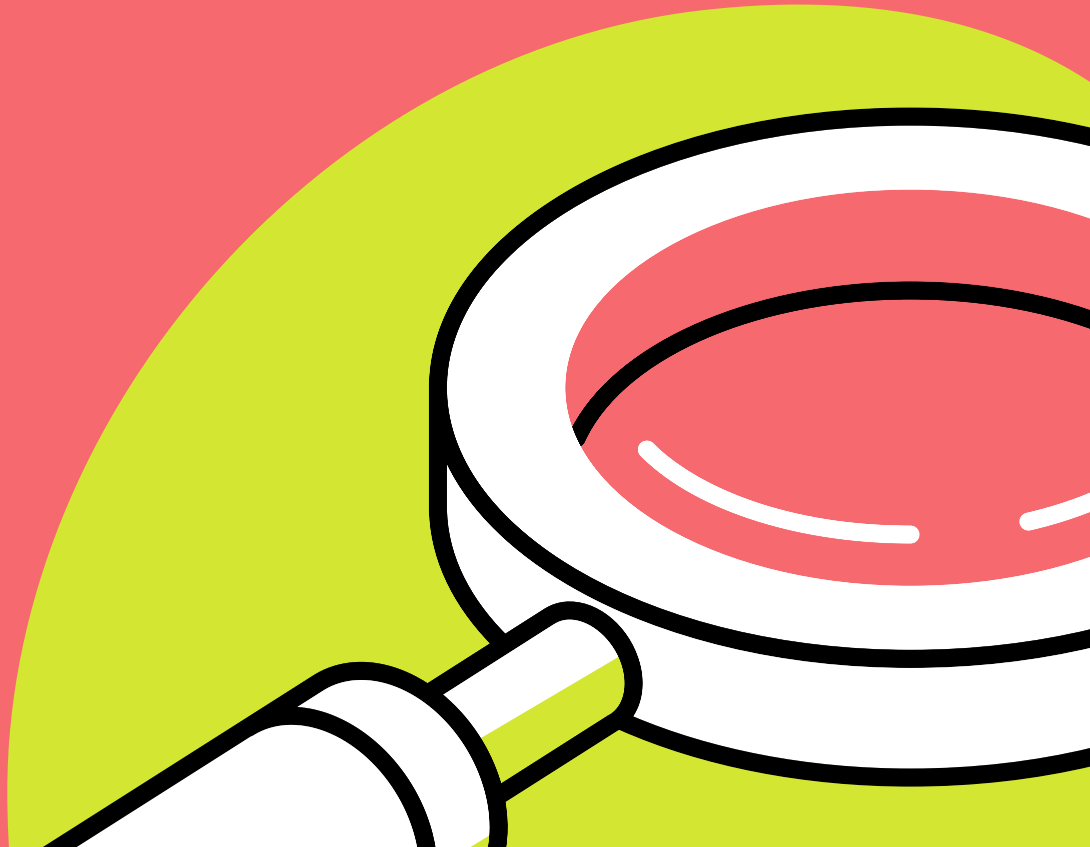
Forked from Databases_Projects / CS_A1155 Summer 2023 / Project Vaccine Distribution
1 commit behind, 5 commits ahead of the upstream repository.

Create merge request

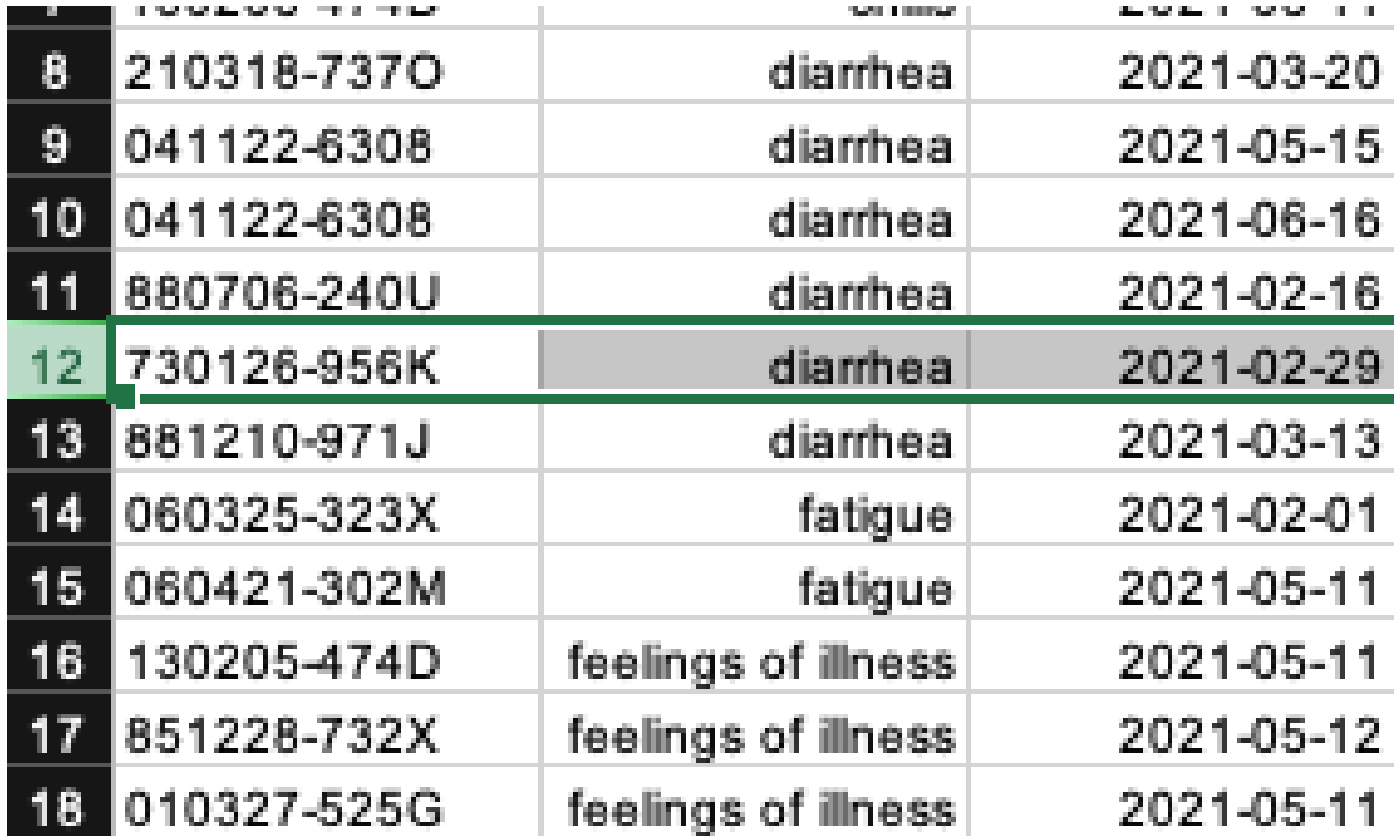
Name	Last commit	Last update
..		
DBprojectSQLcreate.sql	python file	1 week ago
queries.sql	update query final	1 week ago
requirements.txt	python file	1 week ago
test_postgresql_conn.py	upload queries file	1 week ago

Accessibility statement | Saavutettavuusseloste | Privacy notice | Tietosuojailmoitus

Appendix



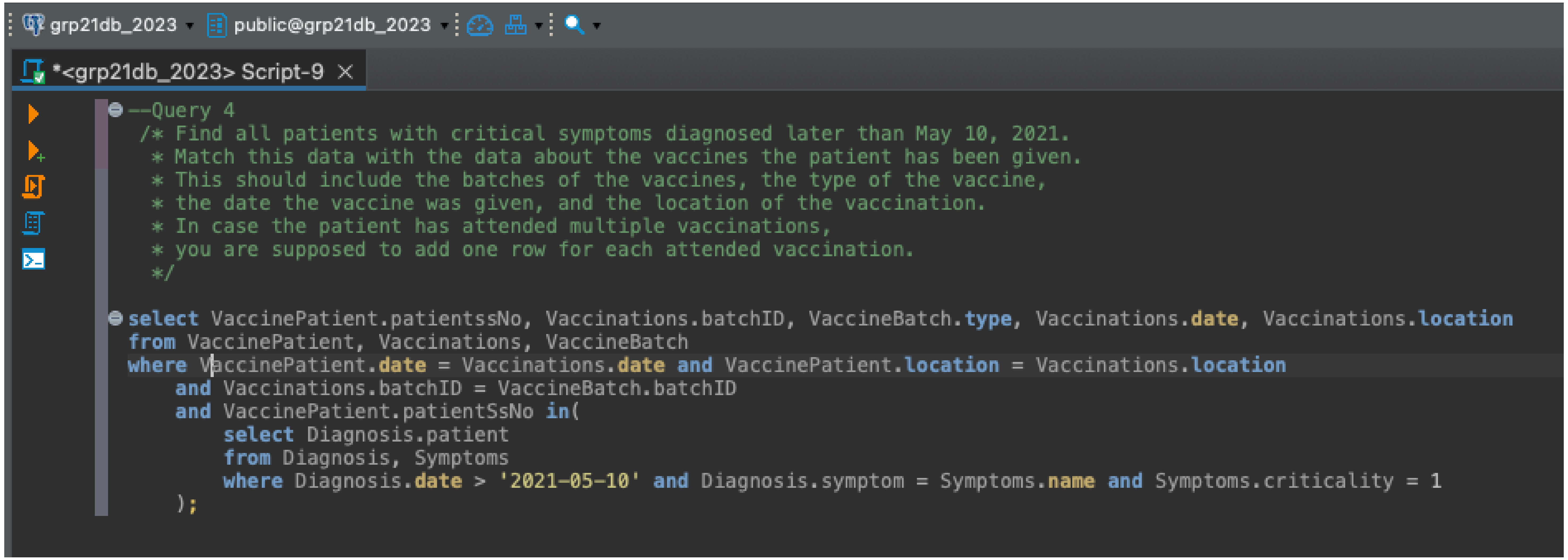
Discoveries



[illegible]	[illegible]	[illegible]	[illegible]
8	210318-737O	diarrhea	2021-03-20
9	041122-6308	diarrhea	2021-05-15
10	041122-6308	diarrhea	2021-06-16
11	880706-240U	diarrhea	2021-02-16
12	730126-956K	diarrhea	2021-02-29
13	881210-971J	diarrhea	2021-03-13
14	060325-323X	fatigue	2021-02-01
15	060421-302M	fatigue	2021-05-11
16	130205-474D	feelings of illness	2021-05-11
17	851228-732X	feelings of illness	2021-05-12
18	010327-525G	feelings of illness	2021-05-11

Discoveries

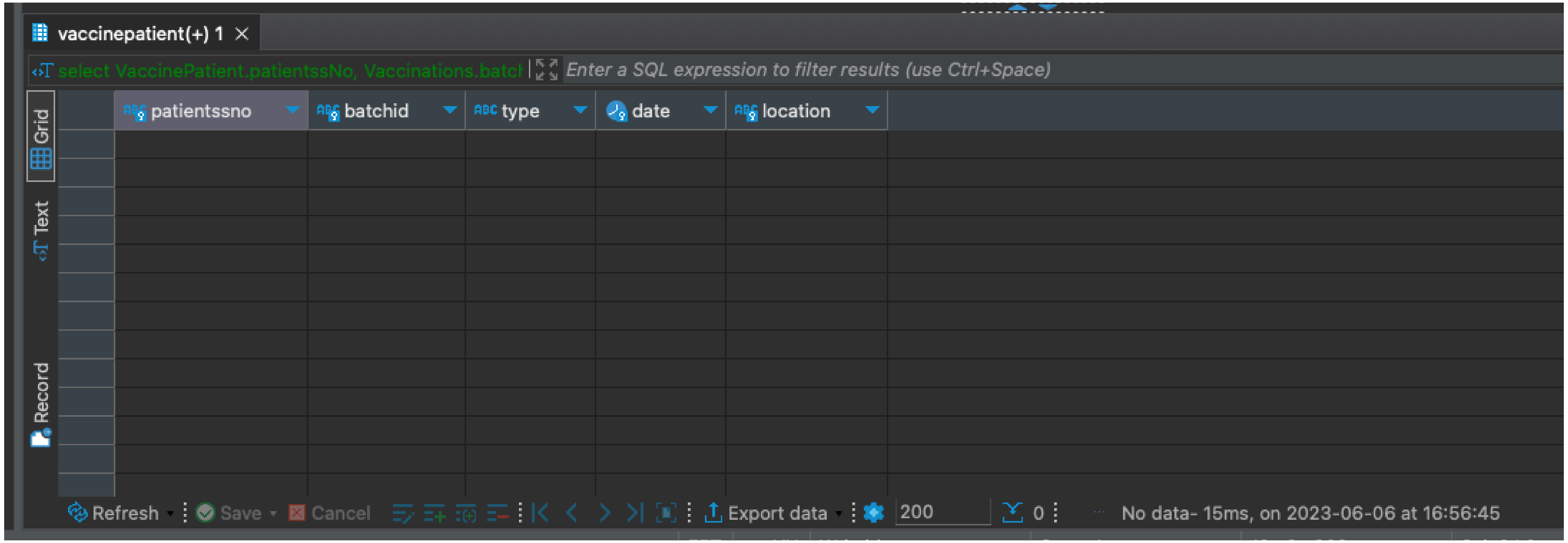
Query:



```
--Query 4
 /* Find all patients with critical symptoms diagnosed later than May 10, 2021.
  * Match this data with the data about the vaccines the patient has been given.
  * This should include the batches of the vaccines, the type of the vaccine,
  * the date the vaccine was given, and the location of the vaccination.
  * In case the patient has attended multiple vaccinations,
  * you are supposed to add one row for each attended vaccination.
  */

select VaccinePatient.patientssNo, Vaccinations.batchID, VaccineBatch.type, Vaccinations.date, Vaccinations.location
from VaccinePatient, Vaccinations, VaccineBatch
where VaccinePatient.date = Vaccinations.date and VaccinePatient.location = Vaccinations.location
    and Vaccinations.batchID = VaccineBatch.batchID
    and VaccinePatient.patientSsNo in(
        select Diagnosis.patient
        from Diagnosis, Symptoms
        where Diagnosis.date > '2021-05-10' and Diagnosis.symptom = Symptoms.name and Symptoms.criticality = 1
    );
```

Discoveries

Result:



vaccinepatient(+) 1

select VaccinePatient.patientssNo, Vaccinations.batcl

Enter a SQL expression to filter results (use Ctrl+Space)

patientssno	batchid	type	date	location

No data- 15ms, on 2023-06-06 at 16:56:45

SQL codes for creating tables

```
create table Manufacturer (
 ID VARCHAR(255) not null primary key,
 country VARCHAR(255) not null,
 phone VARCHAR(255) not null,
 vaccine VARCHAR(255) not null
);

create table VaccineType(
 ID VARCHAR(255) not null primary key,
 name VARCHAR(255) not null,
 doses INT not null,
 tempMin INT not null,
 tempMax INT not null
);

create table VaccineBatch (
 batchID VARCHAR(255) not null primary key,
 amount INT not null,
 type VARCHAR(255) not null,
 manufacturer VARCHAR(255) not null,
 manufDate DATE not null,
 expiration DATE not null,
 location VARCHAR(255)
);

create table VaccinationStations (
 name VARCHAR(255) not null primary key,
 address VARCHAR(255) not null,
 phone VARCHAR(255) not null
);
```

```
create table TransportationLog (
 batchID VARCHAR(255) not null,
 arrival VARCHAR not NULL,
 departure VARCHAR(255) not null,
 DateArr DATE not null,
 DateDep DATE not null,
 primary key (batchID,arrival,DateArr)
);

create table Shifts (
 station VARCHAR (255) not NULL,
 weekday VARCHAR (255) not null check (weekday in
('Monday','Tuesday','Wednesday','Thursday','Friday'
)),
 worker VARCHAR(255) not null,
 primary key(station, weekday, worker)
);

create table StaffMembers (
 "social security number" VARCHAR(255) not null
primary key,
 name VARCHAR(255) not null,
 "date of birth" DATE not null,
 phone VARCHAR(255),
 role VARCHAR(255) not NULL,
 "vaccination status" int not null,
 hospital VARCHAR(255) not NULL
);
```

```
create table Vaccinations (
 "date" DATE not null ,
 location VARCHAR(255) not null,
 batchID VARCHAR(255) not null,
 primary key ("date",location, batchID)
);

create table Patient (
 ssNo VARCHAR(255) not null primary key,
 name VARCHAR(255) not null,
 "date of birth" DATE not null,
 gender VARCHAR(1) not NULL check (gender in
('M','F') )
);

create table VaccinePatient (
 patientSsNo VARCHAR(255) not null,
 "date" DATE not null,
 location VARCHAR(255) not null,
 primary key (patientSsNo, "date",location)
);

create table Symptoms(
 name VARCHAR(255) not null primary key,
 criticality int not null
);

create table Diagnosis (
 patient VARCHAR(255) not null,
 symptom VARCHAR(255) not null,
 "date" DATE not null,
 primary key(patient,symptom,"date")
);
```

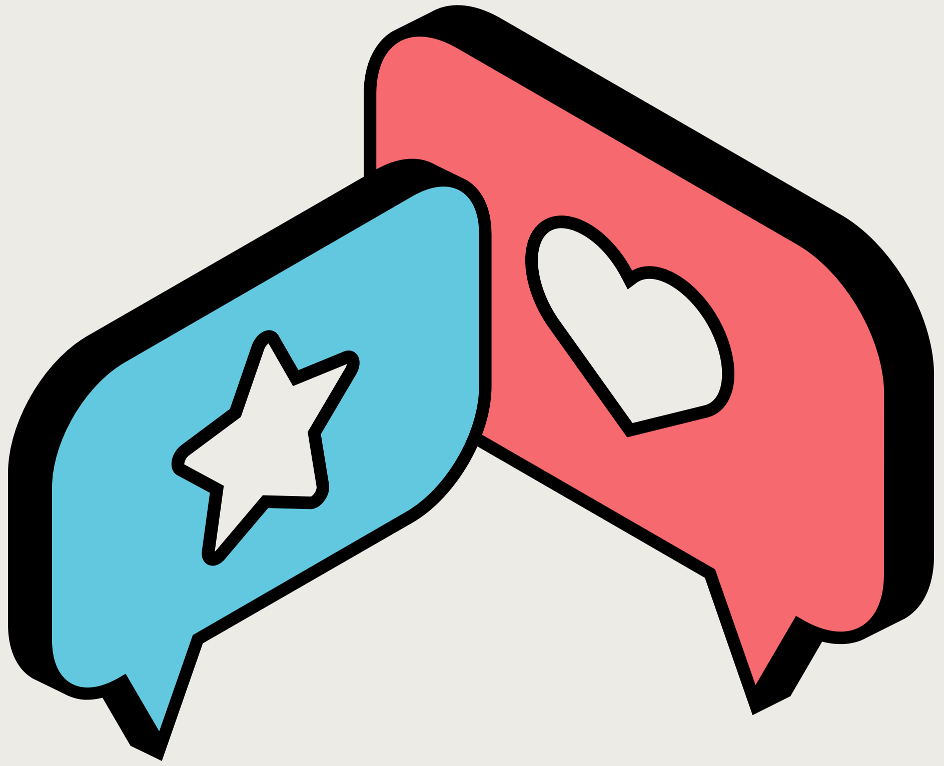
SQL codes for creating tables (cont.)

```
-- Add foreign keys
alter table VaccineBatch
add constraint FK_manufacturer
foreign key (manufacturer) references
Manufacturer(ID) ON DELETE cascade on update
cascade;

alter table VaccineBatch
add constraint FK_type
foreign key (type) references VaccineType(ID)
ON DELETE cascade on update cascade;

alter table TransportationLog
add constraint FK_batchID_log
foreign key (batchID) references
VaccineBatch(batchID) ON DELETE cascade on
update cascade;

alter table TransportationLog
add constraint FK_arrival_log
foreign key (arrival) references
VaccinationStations(name) ON DELETE cascade
on update cascade;

alter table TransportationLog
add constraint FK_departure_log
foreign key (departure) references
VaccinationStations(name) ON DELETE cascade
on update cascade;

alter table Shifts
add constraint FK_station
foreign key (station) references
VaccinationStations(name) ON DELETE cascade on update
cascade;

alter table Shifts
add constraint FK_worker
foreign key (worker) references StaffMembers("social
security number") ON DELETE cascade on update
cascade;

alter table StaffMembers
add constraint FK_hospital
foreign key (hospital) references
VaccinationStations(name) ON DELETE cascade on update
cascade;

alter table Vaccinations
add constraint FK_location_vac
foreign key (location) references
VaccinationStations(name) ON DELETE cascade on update
cascade;

alter table Vaccinations
add constraint FK_batchID_vac
foreign key (batchID) references
VaccineBatch(batchID) ON DELETE cascade on update
cascade;

alter table VaccinePatient
add constraint FK_patientSsNo
foreign key (patientSsNo) references
Patient(ssNo) ON DELETE cascade on update
cascade;

alter table VaccinePatient
add constraint FK_location_vacPatient
foreign key (location) references
VaccinationStations(name) ON DELETE
cascade on update cascade;

alter table Diagnosis
add constraint FK_patient
foreign key (patient) references
Patient(ssNo) ON DELETE cascade on update
cascade;

alter table Diagnosis
add constraint FK_symptom
foreign key (symptom) references
Symptoms(name) ON DELETE cascade on
update cascade;
```

SQL codes for creating queries

```
--Query 1
with vaccineOnMay10Location as(
select location
from vaccinations v
where date = '2021-05-10')

select "social security number", name,
phone, role, "vaccination status" ,
location
from vaccineOnMay10Location join shifts
on (location = station) join
staffmembers on (worker = "social
security number")
where weekday = 'Monday';

--Query 2
select *
from StaffMembers
where role = 'doctor' and "social
security number" in (select worker from
Shifts where weekday = 'Wednesday')
 and hospital in (select name from
VaccinationStations where address like
'%HELSINKI');

--Query 3
select location, arrival
from VaccineBatch right JOIN
TransportationLog on
VaccineBatch.batchID =
TransportationLog.batchID;

select vaccinebatch.batchID,
VaccinationStations.phone
from VaccineBatch right JOIN TransportationLog on
VaccineBatch.batchID = TransportationLog.batchID,
VaccinationStations
where arrival = VaccinationStations.name and
location <> arrival;

--Query 4
select VaccinePatient.patientssNo,
Vaccinations.batchID, VaccineBatch.type,
Vaccinations.date, Vaccinations.location
from VaccinePatient, Vaccinations, VaccineBatch
where VaccinePatient.date = Vaccinations.date and
VaccinePatient.location = Vaccinations.location
 and Vaccinations.batchID = VaccineBatch.batchID
 and VaccinePatient.patientSsNo in(
  select Diagnosis.patient
  from Diagnosis, Symptoms
  where Diagnosis.date > '2021-05-10' and
Diagnosis.symptom = Symptoms.name and
Symptoms.criticality = 1
 );

--Query 5
create or REPLACE VIEW query5 AS
SELECT ssNo,name,"date of birth",gender, case WHEN
COUNT(ssNo) >= 2 THEN 1 ELSE 0 END AS
vaccinationStatus
FROM Patient JOIN VaccinePatient ON Patient.ssNo =
VaccinePatient.patientSsNo
GROUP BY ssNo;

select * from query5;

--Query 6
select location,  location as
"location/type", sum(amount)
from VaccineBatch
group by location
union
select location,  type as "location/type",
sum(amount)
from VaccineBatch
group by location, type
order by location, "location/type";

--Query 7
SELECT VaccineBatch.TYPE, Diagnosis.symptom,
COUNT(DISTINCT
Diagnosis.Patient)/COUNT(DISTINCT
VaccinePatient.ssNo) AS average_frequency
FROM Diagnosis, VaccinePatients,
Vaccinations, VaccineBatch
WHERE Diagnosis.patient =
VaccinePatients.patientSsNo
AND VaccinePatients.date < Diagnosis.date
AND VaccinePatients.date = Vaccinations.date
AND VaccinePatients.location =
Vaccinations.location
AND Vaccinations.batchID =
VaccineBatch.batchID
GROUP BY VaccineBatch.type,
Diagnosis.symptom;
```

Questions?



Thank you.