Vaccine Distribution - Project Part 2

Group 5

Huiyun Miao 722553

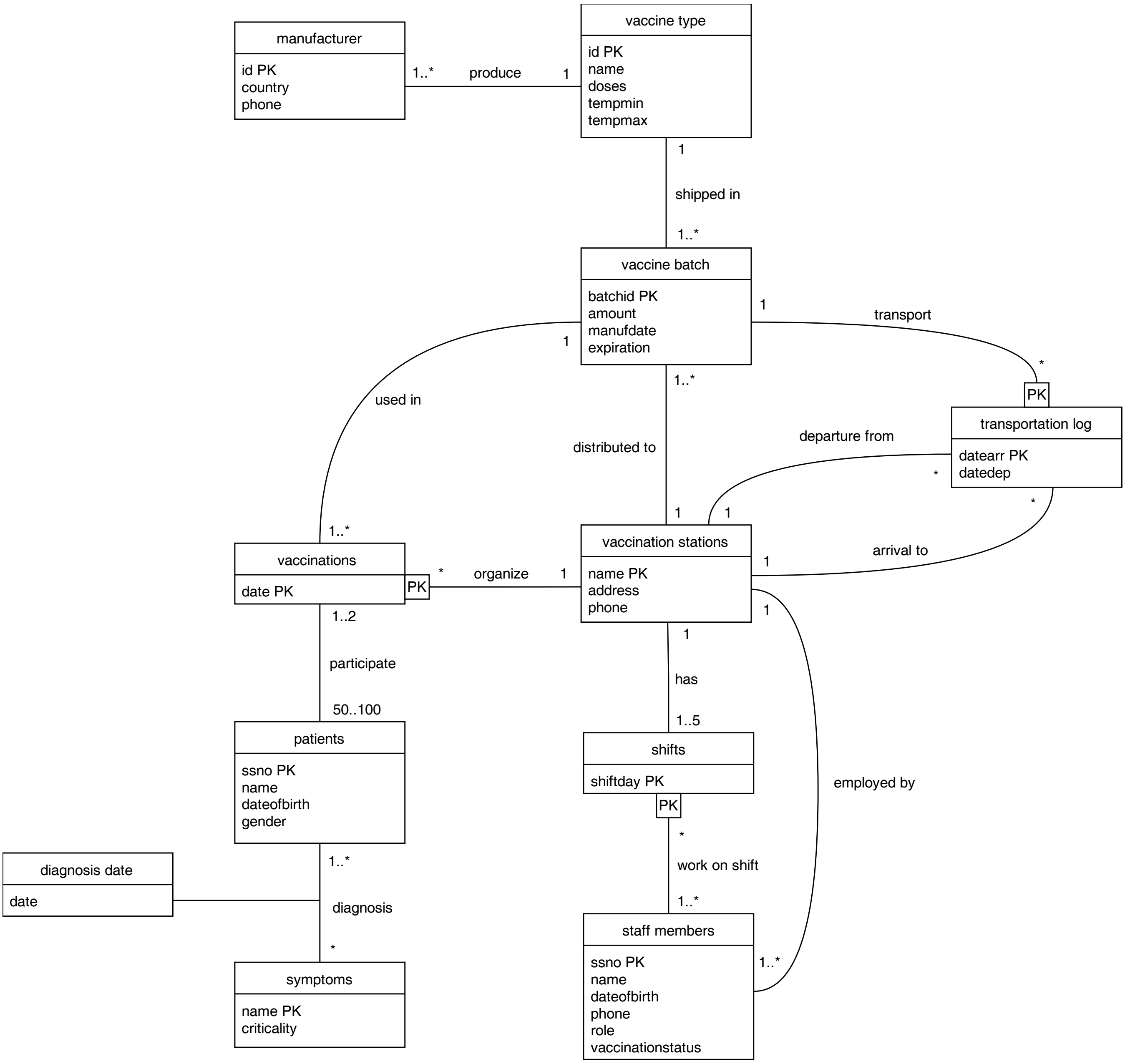
Phuong Ngo 1006184

Linh Tran 1022838

Phuong Anh Tran 1023413

1. Updated UML diagram & relational model

UML diagram:



Relational model:

- manufacturer (id, country, phone, vaccine)
- vaccinetype (id, name, doses, tempmin, tempmax)
- vaccinebatch (batchid, amount, type, manufacturer, manufdate, expiration, loaction)
- vaccinationstations (name, address, phone)
- transportationlog (batchid, datearr, datedep, arrival, departure)
- staffmembers (ssno, name, dateofbirth, phone, role, vaccinationstatus, hospital)
- shifts (ssno, shiftday, station)
- vaccinations (date, location, batchid)
- patients (ssno, name, dateofbirth, gender)
- vaccinepatients (date, ssno, location)
- symptoms (name, criticality)
- diagnosis (patient, symptom, date)

Description:

After reviewing the given data and receiving the feedback for part 1, we find some parts in our UML diagram and relational model that need to update.

To begin with, we update all the names of attribute to be consistent with the names in the excel file. And we fix the mistake we made in part 1: including foreign keys in the UML diagram. The names of the tables and attributes are changed to lowercases because Upper case letters cause problem in PostgreSQL.

Some assumptions we made about the data and the relations in part 1 were not align with the data. Here are the modifications for each table：

- manufacturer:
  According to the data, each manufacturer only has one branch in one county, which means the attribute id can be the primary key instead of both id and country. The attribute vaccine is added to this table.

- vaccinetype:
  The name attribute is added to this table, and the criticalTemperature attribute is divided to tempmin and tempmax.

- Produce:
  This table is dropped because one manufacturer only produces one type of vaccine, and

one type of vaccine can be produced by multiple manufacturers. The relation between vaccinetype and manufacturer is a one-to-many association, not a many-many association as we assumed in part 1.

- vaccinebatch:
  The attribute manufacturer is added to this table, because one type of vaccine could be produced by different manufacturers.

- vaccinationstations:
  Only the name of the table and names of attributes changed.

- transportationlog:
  Initially we added the attribute batchID in this table, but we decided to drop it because each transportation only contains one batch. To identify each transportation, we uses batched and datearr as the primary keys.

- BatchTransport:
  This table is dropped because each transportation only contains one batch.

- staffmembers:
  Only the name of the table and names of attributes changed.

- shifts
  The tables VaccinationShift and WorkOnShift are merged into one table shifts. We find one staff member can work on multiple shifts, so we use ssno and shiftday as the primary keys.

- vaccinations
  The attribute shiftDay is dropped from this table. Because this table includes attribute date, and it is easy to find information on shift day according to the date.

- vaccinepatients
  We only keep date and ssno as the primary keys for this table instead of using all three attributes.

- symptom
  We remove the attribute diagnoseDate because it is more reasonable to include the information of date in the table diagnosis. And we also remove the attribute actionAfterDiagnosis because there is no information about it.

- diagnose
  The attribute date is added to this table.

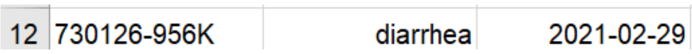
After modifying the relational model, we update the UML diagram accordingly. For the relations in the diagram, we removed the connection between vaccinations and shifts, because the shifts class is connected to vaccinations through vaccination stations. To define the association diagnosis between patients and symptoms, we add diagnosis date in the middle of these classes.

2. Data cleaning

After connecting to the database using SQLAlchemy create_engine, a problem was encountered in reading SQL files for CREATE TABLE and INSERT queries to our designated table. Accordingly, our team decided to resort to option 2 - CREATE TABLE engine connection & fill in tables with Pandas Dataframe to_sql.

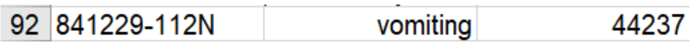
During this process, we need to break down the whole excel file into smaller excel files with each file storing one table only. So, there are a total of 13 smaller excel files. After that, we start populating the data. Since the data type of each attribute is defined when creating tables, we found some errors in the data because the data type is not valid for the attribute. Specifically in the table Diagnosis:

The date entry is not existent as the February of 2021 has only 28 days.



12	730126-956K	diarrhea	2021-02-29

This date entry is not a valid date.



92	841229-112N	vomiting	44237

Therefore, we deleted those erroneous rows before populating the data into the table.

Moreover, as PostgreSQL is case sensitive and we are faced with problems for not being able to create tables despite checking the syntax, we put the name of tables and each column in lowercase. In this case, the code and queries started working.

3. Design choices

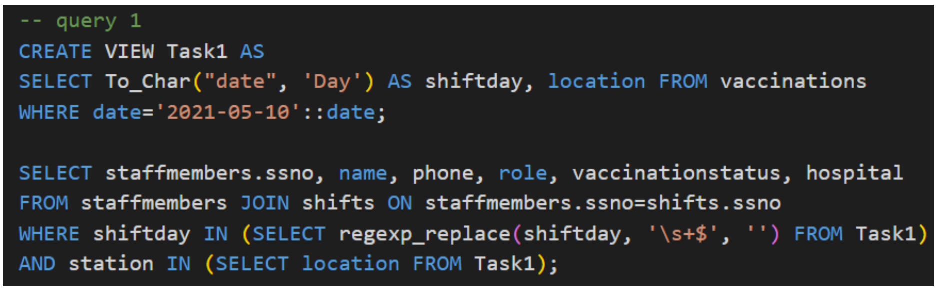
Regarding the data types of the attributes in the table, first, all the input in the table is required to be NOT NULL. For the character types, we chose VARCHAR(n) as it is variable-length with limit and unlike CHAR(n), no padding is done in case the data length is smaller than the value of "n".

With date, the data type we assigned was DATE because only dates without time are included in the table input.

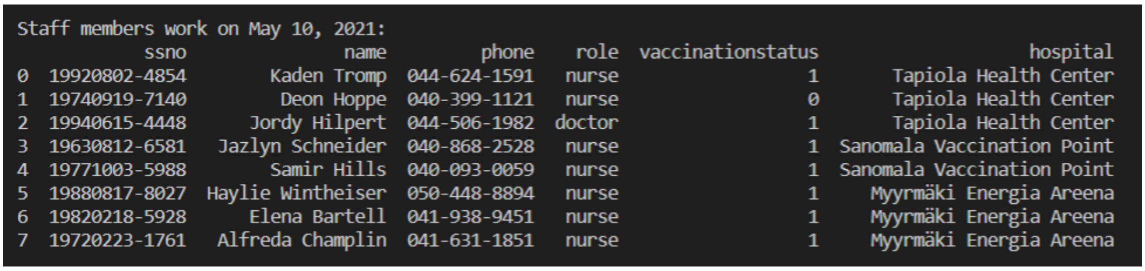
The column doses in the table vaccinetype has INT data types with a check to make sure that each vaccine type has at least 1 and at most 2 doses. The column criticality in the table symptoms has the boolean type with the representation: 1 for True and 0 for False, so we have a check to ensure the value of this column would be 0 or 1.

For 7 required queries, here follows the logic behind them.

(1) In the first query, to find all the staff members working in a vaccination on May 10, 2021, we extracted weekdays from date '2021-05-10' and then joined tables to find out *location* and *shiftday* of the given vaccination. Lastly, we filter *staffmembers* work on that shift.

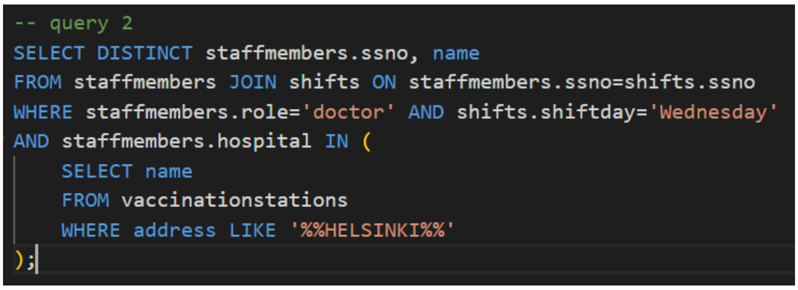


```
-- query 1
CREATE VIEW Task1 AS
SELECT To_Char("date", 'Day') AS shiftday, location FROM vaccinations
WHERE date='2021-05-10'::date;

SELECT staffmembers.ssno, name, phone, role, vaccinationstatus, hospital
FROM staffmembers JOIN shifts ON staffmembers.ssno=shifts.ssno
WHERE shiftday IN (SELECT regexp_replace(shiftday, '\s+$', '') FROM Task1)
AND station IN (SELECT location FROM Task1);
```

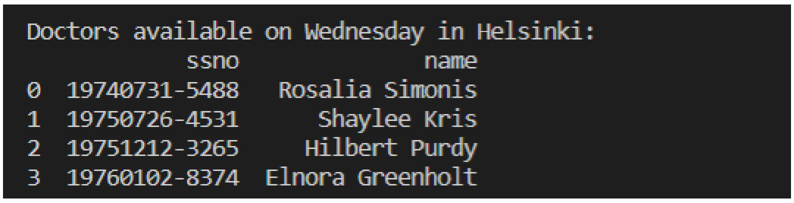


```
Staff members work on May 10, 2021:
            ssno              name         phone    role  vaccinationstatus                 hospital
0  19920802-4854        Kaden Tromp  044-624-1591   nurse                  1     Tapiola Health Center
1  19740919-7140        Deon Hoppe  040-399-1121   nurse                  0     Tapiola Health Center
2  19940615-4448      Jordy Hilpert  044-506-1982  doctor                  1     Tapiola Health Center
3  19630812-6581  Jazlyn Schneider  040-868-2528   nurse                  1  Sanomala Vaccination Point
4  19771003-5988        Samir Hills  040-093-0059   nurse                  1  Sanomala Vaccination Point
5  19880817-8027 Haylie Wintheiser  050-448-8894   nurse                  1      Myyrmäki Energia Areena
6  19820218-5928      Elena Bartell  041-938-9451   nurse                  1      Myyrmäki Energia Areena
7  19720223-1761   Alfreda Champlin  041-631-1851   nurse                  1      Myyrmäki Energia Areena
```

(2) In the second query, to list all the doctors who would be available on Wednesdays in Helsinki, we joined tables to obtain *staffmembers*’ information and their shifts, then filtered hospitals whose location is in Helsinki and *staffmembers* whose role is doctor, hospital is in Helsinki and shift is on Wednesday.



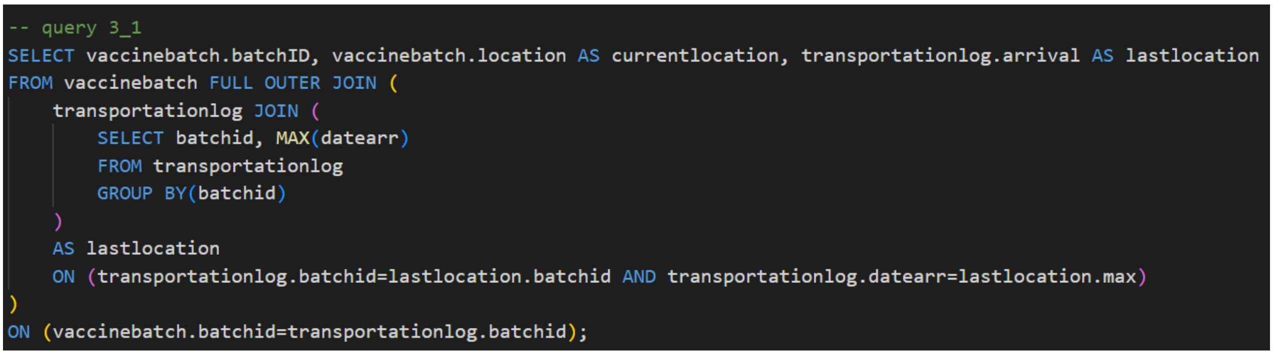
```sql
-- query 2
SELECT DISTINCT staffmembers.ssno, name
FROM staffmembers JOIN shifts ON staffmembers.ssno=shifts.ssno
WHERE staffmembers.role='doctor' AND shifts.shiftday='Wednesday'
AND staffmembers.hospital IN (
    SELECT name
    FROM vaccinationstations
    WHERE address LIKE '%%HELSINKI%%'
);
```



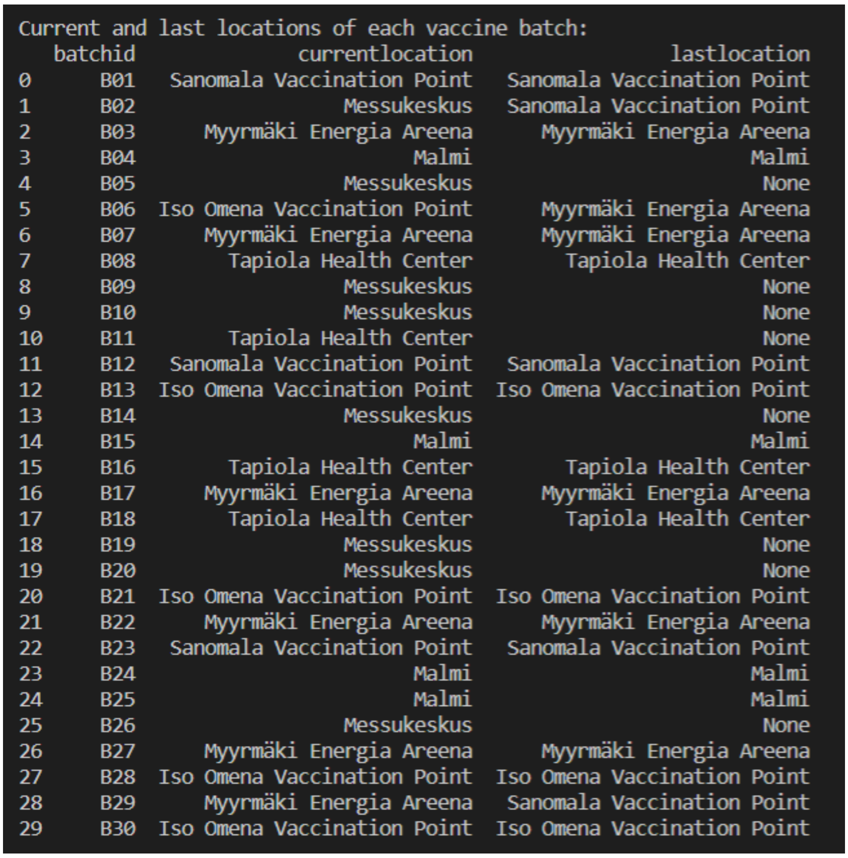
```
Doctors available on Wednesday in Helsinki:
            ssno              name
0  19740731-5488   Rosalia Simonis
1  19750726-4531      Shaylee Kris
2  19751212-3265     Hilbert Purdy
3  19760102-8374  Elnora Greenholt
```

(3) In the third query, our first step is to join tables to obtain vaccine batches' ID, current location and last location in *transportationlog*. Next, we filtered vaccine batches whose current location is not last location in *transportationlog* and join tables to obtain required information.

Query 3_1:

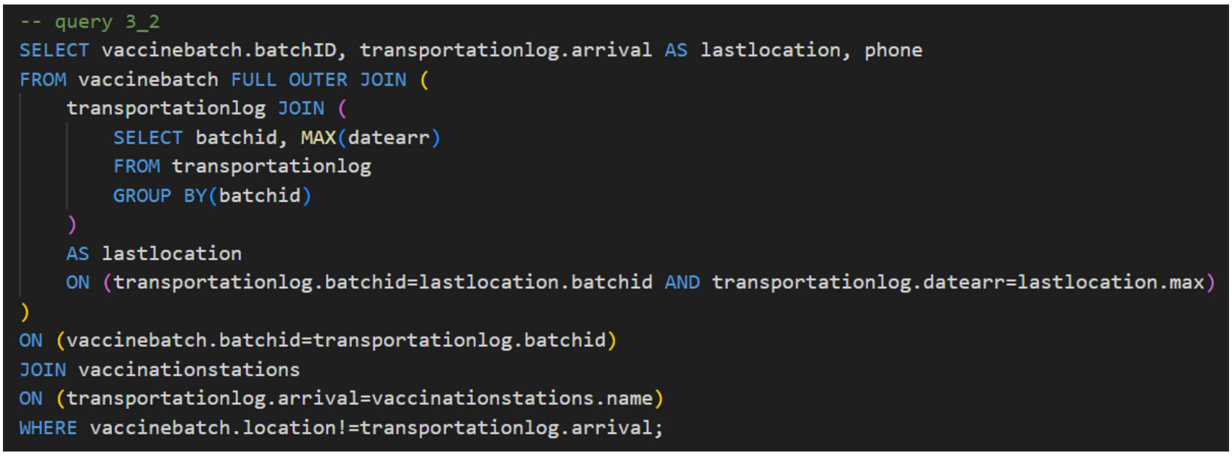


```
-- query 3_1
SELECT vaccinebatch.batchID, vaccinebatch.location AS currentlocation, transportationlog.arrival AS lastlocation
FROM vaccinebatch FULL OUTER JOIN (
    transportationlog JOIN (
        SELECT batchid, MAX(datearr)
        FROM transportationlog
        GROUP BY(batchid)
    )
    AS lastlocation
    ON (transportationlog.batchid=lastlocation.batchid AND transportationlog.datearr=lastlocation.max)
)
ON (vaccinebatch.batchid=transportationlog.batchid);
```

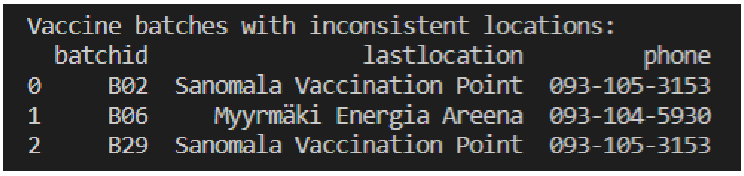


```
Current and last locations of each vaccine batch:
   batchid            currentlocation              lastlocation
0      B01  Sanomala Vaccination Point  Sanomala Vaccination Point
1      B02                 Messukeskus  Sanomala Vaccination Point
2      B03       Myyrmäki Energia Areena       Myyrmäki Energia Areena
3      B04                       Malmi                        Malmi
4      B05                 Messukeskus                         None
5      B06  Iso Omena Vaccination Point       Myyrmäki Energia Areena
6      B07       Myyrmäki Energia Areena       Myyrmäki Energia Areena
7      B08        Tapiola Health Center        Tapiola Health Center
8      B09                 Messukeskus                         None
9      B10                 Messukeskus                         None
10     B11        Tapiola Health Center                         None
11     B12  Sanomala Vaccination Point  Sanomala Vaccination Point
12     B13  Iso Omena Vaccination Point  Iso Omena Vaccination Point
13     B14                 Messukeskus                         None
14     B15                       Malmi                        Malmi
15     B16        Tapiola Health Center        Tapiola Health Center
16     B17       Myyrmäki Energia Areena       Myyrmäki Energia Areena
17     B18        Tapiola Health Center        Tapiola Health Center
18     B19                 Messukeskus                         None
19     B20                 Messukeskus                         None
20     B21  Iso Omena Vaccination Point  Iso Omena Vaccination Point
21     B22       Myyrmäki Energia Areena       Myyrmäki Energia Areena
22     B23  Sanomala Vaccination Point  Sanomala Vaccination Point
23     B24                       Malmi                        Malmi
24     B25                       Malmi                        Malmi
25     B26                 Messukeskus                         None
26     B27       Myyrmäki Energia Areena       Myyrmäki Energia Areena
27     B28  Iso Omena Vaccination Point  Iso Omena Vaccination Point
28     B29       Myyrmäki Energia Areena  Sanomala Vaccination Point
29     B30  Iso Omena Vaccination Point  Iso Omena Vaccination Point
```

Query 3_2:

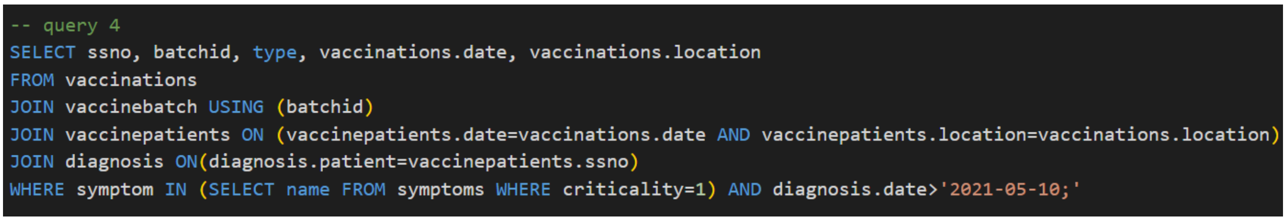


```
-- query 3_2
SELECT vaccinebatch.batchID, transportationlog.arrival AS lastlocation, phone
FROM vaccinebatch FULL OUTER JOIN (
    transportationlog JOIN (
        SELECT batchid, MAX(datearr)
        FROM transportationlog
        GROUP BY(batchid)
    )
    AS lastlocation
    ON (transportationlog.batchid=lastlocation.batchid AND transportationlog.datearr=lastlocation.max)
)
ON (vaccinebatch.batchid=transportationlog.batchid)
JOIN vaccinationstations
ON (transportationlog.arrival=vaccinationstations.name)
WHERE vaccinebatch.location!=transportationlog.arrival;
```

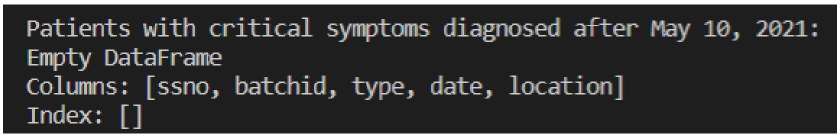


```
Vaccine batches with inconsistent locations:
  batchid                lastlocation         phone
0     B02  Sanomala Vaccination Point  093-105-3153
1     B06     Myyrmäki Energia Areena  093-104-5930
2     B29  Sanomala Vaccination Point  093-105-3153
```

(4) In the fourth query, we are required to find all patients with critical symptoms diagnosed later than May 10, 2021 and match this data with the data about the vaccines the patient has been given, namely the batches of the vaccines, the type of the vaccine, the date the vaccine was given, and the location of the vaccination. We first filtered symptoms which are critical and then patients whose diagnosis is after ‘2021-05-10’ and their symptoms are in the critical symptoms list.

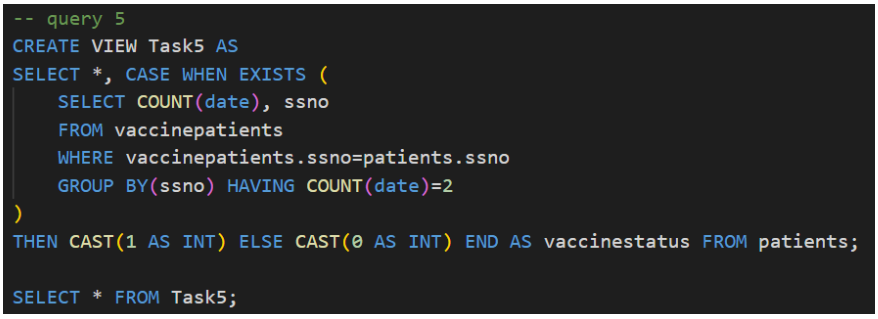


```sql
-- query 4
SELECT ssno, batchid, type, vaccinations.date, vaccinations.location
FROM vaccinations
JOIN vaccinebatch USING (batchid)
JOIN vaccinepatients ON (vaccinepatients.date=vaccinations.date AND vaccinepatients.location=vaccinations.location)
JOIN diagnosis ON(diagnosis.patient=vaccinepatients.ssno)
WHERE symptom IN (SELECT name FROM symptoms WHERE criticality=1) AND diagnosis.date>'2021-05-10;'
```

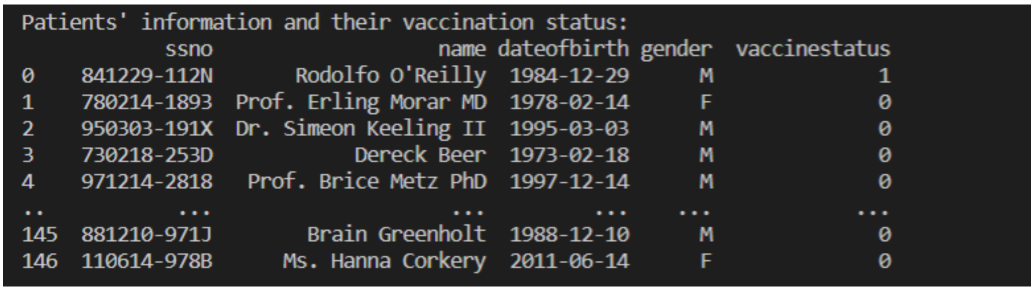


```
Patients with critical symptoms diagnosed after May 10, 2021:
Empty DataFrame
Columns: [ssno, batchid, type, date, location]
Index: []
```

(5) In the fifth query, it is required of a view for patients with additional column *vaccinationstatus*. As we notice that all vaccines require 2 doses, we first filtered patients who have attended vaccinations twice. Afterwards, we created a view from table ‘patients’ with an additional column *vaccinestatus* if this patient appears in the patient list above.

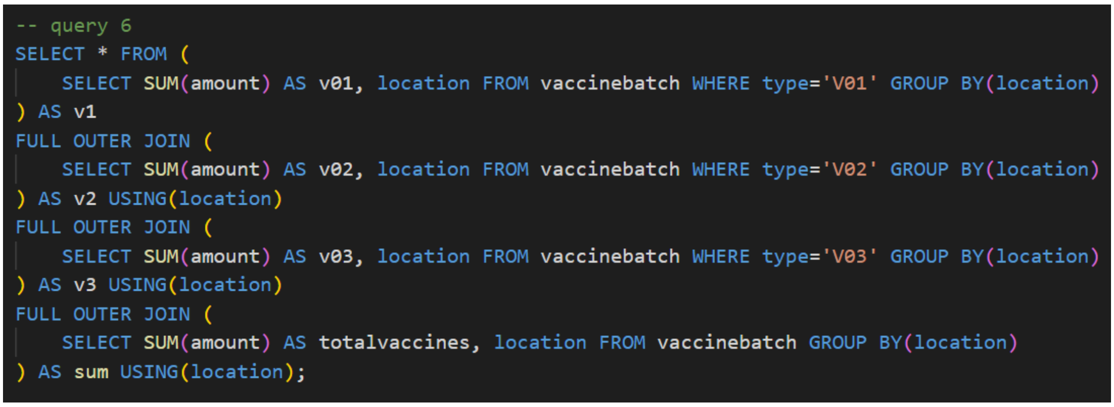


```
-- query 5
CREATE VIEW Task5 AS
SELECT *, CASE WHEN EXISTS (
    SELECT COUNT(date), ssno
    FROM vaccinepatients
    WHERE vaccinepatients.ssno=patients.ssno
    GROUP BY(ssno) HAVING COUNT(date)=2
)
THEN CAST(1 AS INT) ELSE CAST(0 AS INT) END AS vaccinestatus FROM patients;

SELECT * FROM Task5;
```

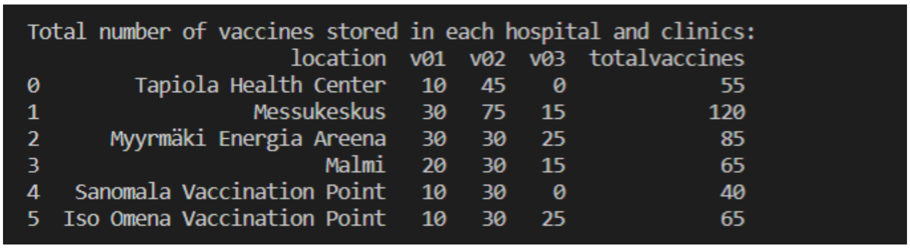


```
Patients' information and their vaccination status:
            ssno                  name dateofbirth gender  vaccinestatus
0    841229-112N       Rodolfo O'Reilly  1984-12-29      M              1
1    780214-1893  Prof. Erling Morar MD  1978-02-14      F              0
2    950303-191X  Dr. Simeon Keeling II  1995-03-03      M              0
3    730218-253D            Dereck Beer  1973-02-18      M              0
4    971214-2818   Prof. Brice Metz PhD  1997-12-14      M              0
..           ...                   ...         ...    ...            ...
145  881210-971J        Brain Greenholt  1988-12-10      M              0
146  110614-978B      Ms. Hanna Corkery  2011-06-14      F              0
```

(6) In the sixth query, to find the total number of vaccines stored in each hospital and clinic, for each hospital, we calculated the total number of each vaccine and joined these tables to obtain all the total numbers of each vaccine. In the previously obtained table, we grouped by hospital to obtain the total number of all vaccines.

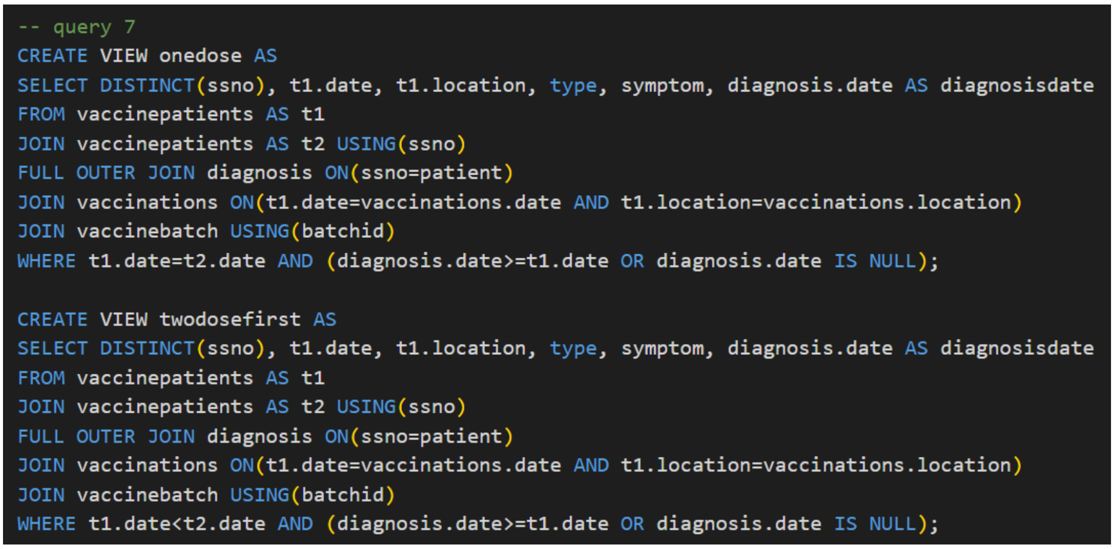


```
-- query 6
SELECT * FROM (
    SELECT SUM(amount) AS v01, location FROM vaccinebatch WHERE type='V01' GROUP BY(location)
) AS v1
FULL OUTER JOIN (
    SELECT SUM(amount) AS v02, location FROM vaccinebatch WHERE type='V02' GROUP BY(location)
) AS v2 USING(location)
FULL OUTER JOIN (
    SELECT SUM(amount) AS v03, location FROM vaccinebatch WHERE type='V03' GROUP BY(location)
) AS v3 USING(location)
FULL OUTER JOIN (
    SELECT SUM(amount) AS totalvaccines, location FROM vaccinebatch GROUP BY(location)
) AS sum USING(location);
```

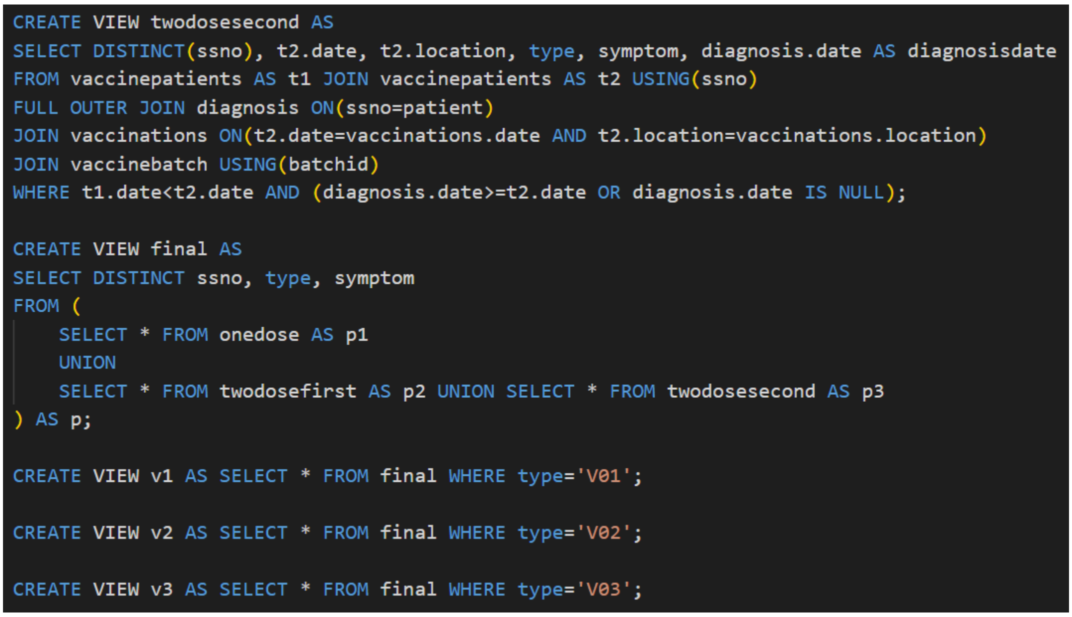


```
Total number of vaccines stored in each hospital and clinics:
                      location  v01  v02  v03  totalvaccines
0        Tapiola Health Center   10   45    0             55
1                  Messukeskus   30   75   15            120
2      Myyrmäki Energia Areena   30   30   25             85
3                        Malmi   20   30   15             65
4     Sanomala Vaccination Point  10   30    0             40
5    Iso Omena Vaccination Point  10   30   25             65
```

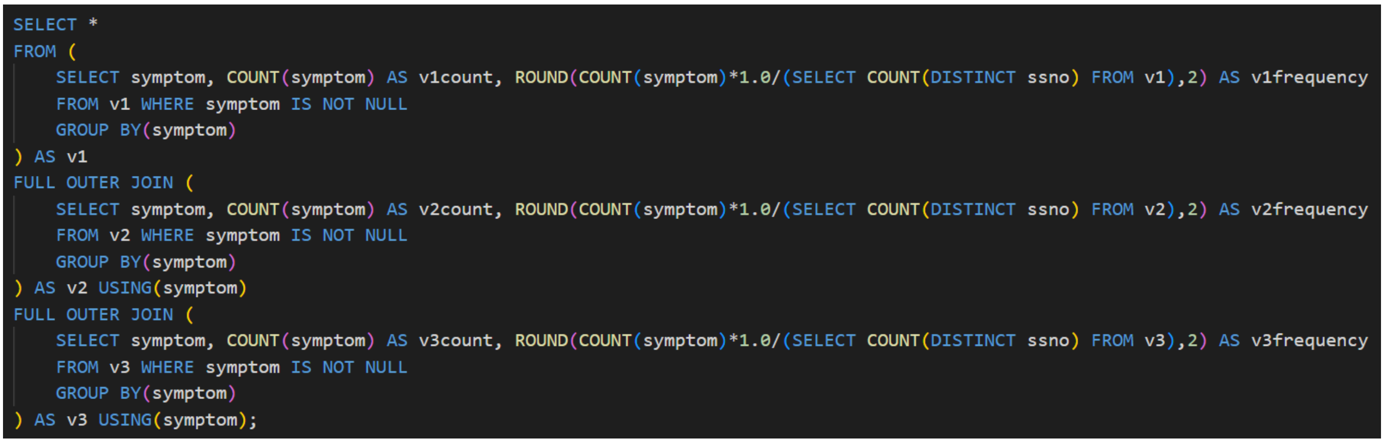
(7) In the seventh query, we are required to calculate the average frequency of different symptoms diagnosed. First, *vaccinepatients* was divided into 2 groups: those who have 1 dose and those who have 2 doses. After that, the group with 2 doses was divided into 2: first dose group and second dose group. For each group, we found patients whose symptoms happened after the vaccination date and created a view including patients' ssno and symptoms, vaccine used for that vaccination, date and location.Then we did the union to the views above and created a new view with *distinct (ssno, type, symptom)*. For each vaccine type, a new view was created and then for each view, the number of times a symptom happened was counted.



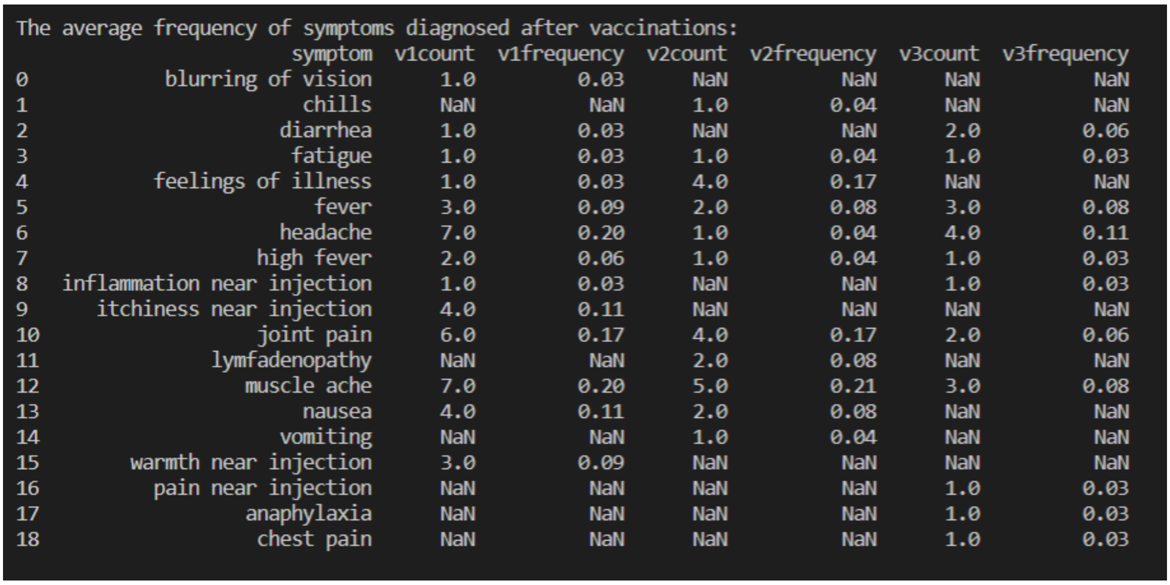
```
-- query 7
CREATE VIEW onedose AS
SELECT DISTINCT(ssno), t1.date, t1.location, type, symptom, diagnosis.date AS diagnosisdate
FROM vaccinepatients AS t1
JOIN vaccinepatients AS t2 USING(ssno)
FULL OUTER JOIN diagnosis ON(ssno=patient)
JOIN vaccinations ON(t1.date=vaccinations.date AND t1.location=vaccinations.location)
JOIN vaccinebatch USING(batchid)
WHERE t1.date=t2.date AND (diagnosis.date>=t1.date OR diagnosis.date IS NULL);

CREATE VIEW twodosefirst AS
SELECT DISTINCT(ssno), t1.date, t1.location, type, symptom, diagnosis.date AS diagnosisdate
FROM vaccinepatients AS t1
JOIN vaccinepatients AS t2 USING(ssno)
FULL OUTER JOIN diagnosis ON(ssno=patient)
JOIN vaccinations ON(t1.date=vaccinations.date AND t1.location=vaccinations.location)
JOIN vaccinebatch USING(batchid)
WHERE t1.date<t2.date AND (diagnosis.date>=t1.date OR diagnosis.date IS NULL);
```



```
CREATE VIEW twodosesecond AS
SELECT DISTINCT(ssno), t2.date, t2.location, type, symptom, diagnosis.date AS diagnosisdate
FROM vaccinepatients AS t1 JOIN vaccinepatients AS t2 USING(ssno)
FULL OUTER JOIN diagnosis ON(ssno=patient)
JOIN vaccinations ON(t2.date=vaccinations.date AND t2.location=vaccinations.location)
JOIN vaccinebatch USING(batchid)
WHERE t1.date<t2.date AND (diagnosis.date>=t2.date OR diagnosis.date IS NULL);

CREATE VIEW final AS
SELECT DISTINCT ssno, type, symptom
FROM (
    SELECT * FROM onedose AS p1
    UNION
    SELECT * FROM twodosefirst AS p2 UNION SELECT * FROM twodosesecond AS p3
) AS p;

CREATE VIEW v1 AS SELECT * FROM final WHERE type='V01';

CREATE VIEW v2 AS SELECT * FROM final WHERE type='V02';

CREATE VIEW v3 AS SELECT * FROM final WHERE type='V03';
```



```
SELECT *
FROM (
    SELECT symptom, COUNT(symptom) AS v1count, ROUND(COUNT(symptom)*1.0/(SELECT COUNT(DISTINCT ssno) FROM v1),2) AS v1frequency
    FROM v1 WHERE symptom IS NOT NULL
    GROUP BY(symptom)
) AS v1
FULL OUTER JOIN (
    SELECT symptom, COUNT(symptom) AS v2count, ROUND(COUNT(symptom)*1.0/(SELECT COUNT(DISTINCT ssno) FROM v2),2) AS v2frequency
    FROM v2 WHERE symptom IS NOT NULL
    GROUP BY(symptom)
) AS v2 USING(symptom)
FULL OUTER JOIN (
    SELECT symptom, COUNT(symptom) AS v3count, ROUND(COUNT(symptom)*1.0/(SELECT COUNT(DISTINCT ssno) FROM v3),2) AS v3frequency
    FROM v3 WHERE symptom IS NOT NULL
    GROUP BY(symptom)
) AS v3 USING(symptom);
```



```
The average frequency of symptoms diagnosed after vaccinations:
                       symptom  v1count  v1frequency  v2count  v2frequency  v3count  v3frequency
0           blurring of vision      1.0         0.03      NaN          NaN      NaN          NaN
1                       chills      NaN          NaN      1.0         0.04      NaN          NaN
2                     diarrhea      1.0         0.03      NaN          NaN      2.0         0.06
3                      fatigue      1.0         0.03      1.0         0.04      1.0         0.03
4          feelings of illness      1.0         0.03      4.0         0.17      NaN          NaN
5                        fever      3.0         0.09      2.0         0.08      3.0         0.08
6                     headache      7.0         0.20      1.0         0.04      4.0         0.11
7                   high fever      2.0         0.06      1.0         0.04      1.0         0.03
8  inflammation near injection      1.0         0.03      NaN          NaN      1.0         0.03
9     itchiness near injection      4.0         0.11      NaN          NaN      NaN          NaN
10                  joint pain      6.0         0.17      4.0         0.17      2.0         0.06
11              lymfadenopathy      NaN          NaN      2.0         0.08      NaN          NaN
12                 muscle ache      7.0         0.20      5.0         0.21      3.0         0.08
13                      nausea      4.0         0.11      2.0         0.08      NaN          NaN
14                    vomiting      NaN          NaN      1.0         0.04      NaN          NaN
15       warmth near injection      3.0         0.09      NaN          NaN      NaN          NaN
16         pain near injection      NaN          NaN      NaN          NaN      1.0         0.03
17                 anaphylaxia      NaN          NaN      NaN          NaN      1.0         0.03
18                  chest pain      NaN          NaN      NaN          NaN      1.0         0.03
```