Vaccine distribution project

Group 2 members

Adilet Beketov
Thang Phan 1008713
Daria Liutina 899622

Part 1. Reasoning behind design choices

General assumptions

Our general assumptions are:

- all the dates are character strings in the format 'year-month-day';
- social security number (ssNo) is unique and determines the rest of the person's data. It has a fixed form of 'YYMMDD-NNNC', where the first 6 digits come from the birthdate, NNN is a number between 002 and 899 and C is the check character which can be a digit or a letter;

Choosing appropriate classes and associations

Analyzing the given description of the database to be built we started with underlining nouns. However, not every noun becomes a class/attribute.

1. Vaccine manufacturers and batches

Our assumptions:

- one manufacturer can only receive one license for producing one type of vaccine so there can be only one vaccine ID associated with the manufacturer's ID;
- manufacturer ID is an 2-character long string and it's unique;
- each manufacturer can only have one phone number and one production country, stored as character strings;
- Vaccine ID is a 3-character long string and it's unique. It can also be referenced to as vaccine type;
- batch ID is a 3-character long string and it's unique;
- Every vaccine type requires 2 doses;
- Critical storage temperature varies between vaccines and is between -90 to +8 degrees Celsius;
- A batch contains at least 10 and at most 20 doses of vaccine.

Used classes and their attributes related to vaccine manufacturers and batches:

- Manufacturer (<u>ID</u>, country, phone, vaccine): manufacturer's ID is the primary key as it's unique and is enough for determining the rest of the attributes of a manufacturer.

- VaccineBatch (batchID, amount, manufacturer, manufDate, expiration, location): batch ID is the primary key as it's unique and determines the rest of the attributes. amount is the number of vaccines in a batch and it's a positive integer between 10-20. Type is the VaccineID. manufDate and expiration are the production and expiration dates respectively. Location is the vaccination point the batch is at. The data about vaccine batches are crucial so missing information is not acceptable. Thus there's a constraint that no attribute of the class can be NULL. Vaccine type is determined from the manufacturer
- VaccineType (ID, name, doses, tempMin, tempMax): vaccine ID is unique and describes the properties of a vaccine. Doses is the number of doses required for full protection and is a positive integer number between 1-2. temp is the critical storage temperature (in degrees Celsius) and is an integer number between -90 to 8. Due to the critical importance of the vaccine data no attribute is allowed to have a NULL value.

2. Transportation log

Our assumptions:
- each vaccination station can only have one phone number and one address, stored as strings;
- there might be cases when the same vaccine batch is transported several times, so the departure and arrival dates together with the batch ID and arrival and departure hospitals are needed to determine a transportation event.

Used classes and their attributes related to vaccine logistics:
- VaccinationStation (name, address, phone): name of a hospital is the primary key as it's unique. Address and phone are strings with NULL as their default values.
- TransportationLog (batchID, arrival, departure, dateArr, dateDep): batchID is the ID of the vaccine batch. dateArr and dateDep are the departure and arrival dates respectively. departure and arrival are the departure and arrival hospitals respectively.

3. Vaccination shifts

Our assumptions:
- status of an employee describes whether or not they have a full set of vaccines ('1' if they have it and '0' if they don't, can't be NULL;
- one person can be only one clinic's employee;
- Names are stored as max 30 character strings;
- weekday is a character string and it has one of the following values: 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'. There's no vaccination during the weekend.

Used classes and attributes related to staff:
- StaffMembers (ssNo, name, birthday, phone, role, status, hospital): ssNo is the social security number, which needs to be of a certain form and can't be NULL. It also forms

the primary key. Name and birthday are character strings that can't be NULL. Status is either '0' or '1', the default value is 0. Role is a string which has to be either 'nurse' or 'doctor' (or NULL), the default value is NULL. Hospital is the vaccination station the person is employed at.

Used associations:

- Shifts (<u>station</u>, <u>weekday</u>, <u>worker</u>): station is the name of the hospital. Weekday is the day for which the shift plan is made. Worker is the employee's social security number. Together the attributes form the key for this class.

4. Vaccination events

Our assumptions:

- vaccine batches can't be split between hospitals

Used associations for vaccination events:

- Vaccinations (<u>date</u>, <u>location</u>, <u>batchID</u>): date is the vaccination event date. Location is the vaccination station. batchID is the ID of the vaccine batch used in the event. Together date, location and batchID form the primary key.

5. Patients and symptoms

Our assumptions:

- the social security number, name, birthday and status of patients have the same properties as in the employee-class;
- a patient might report a symptom several times;
- a patient only attends one vaccination event per day
- A patient is fully vaccinated (status = 1) if they have received at least two doses of vaccine (regardless of the vaccine type, i.e two doses of different vaccines will mean full vaccination).

Used classes and attributes related to the patients and symptoms:

- Patients (<u>ssNo</u>, name, birthday, gender, status): gender is a character and is either 'M' or 'F', status describes if the patient is fully vaccinated ('1') or not ('0').
- Symptoms (<u>name</u>, criticality): name is a string of max 50 characters, can't be NULL. Criticality is either '1' or '0'.

Used associations:

- VaccinePatients (<u>date</u>, location, <u>ssNo</u>): location is the vaccination station. Date and the patient's social security number give the key for the vaccination case;
- Diagnosis (<u>patient</u>, <u>symptom</u>, <u>date</u>): date is the date of a symptom report. Patient's social security number, date and the name of the symptom form the key for this association.

What are the non-trivial functional dependencies of the database?

- Manufacturer(ID, country, phone, vaccine)
  - ID -> country, phone, vaccine
- VaccineBatch(batchID, amount, type, manufacturer, manufDate, expiration, location)
  - batchID -> amount, type, manufacturer, manufDate, expiration, location
  - manufacturer -> type
  - manufDate -> expiration
  - expiration -> manufDate
- VaccineType(ID, name, doses, tempMin, tempMax)
  - ID -> name, doses, tempMin, tempMax
- VaccinationStation(name, address, phone)
  - name -> address, phone
- TransportationLog(batchID, arrival, departure, dateArr, dateDep)
  - No FDs
- Shifts(station, weekday, worker)
  - No FDs
- StaffMembers(ssNo, name, birthday, phone, role, status, hospital)
  - ssNo -> name, birthday, phone, role, status, hospital
- Vaccinations(date, location, batchID)
  - No FDs
- Patients(ssNo, name, birthday, gender, status)
  - ssNo -> name, birthday, gender, status
- Symptoms (name, criticality)
  - name -> criticality
- VaccinePatients(date, location, ssNo)
  - date, ssNo -> location
- Diagnosis(patient, symptom, date)
  - No FDs

Is the database in the Boyce-Codd Normal Form?

Most relations are in BCNF with an exception of the VaccineBatch relation. Given the original relation VaccineBatch (batchID, amount, type, manufacturer, manufDate, expiration, location) and its functional dependencies, the decomposed result is two relations:

- ExpireInfo(manufDate, expiration)
  - manufDate -> expiration
- VaccineBatch(batchID, amount, manufacturer, manufDate, location)
  - batchID -> amount, manufacturer, manufDate, location

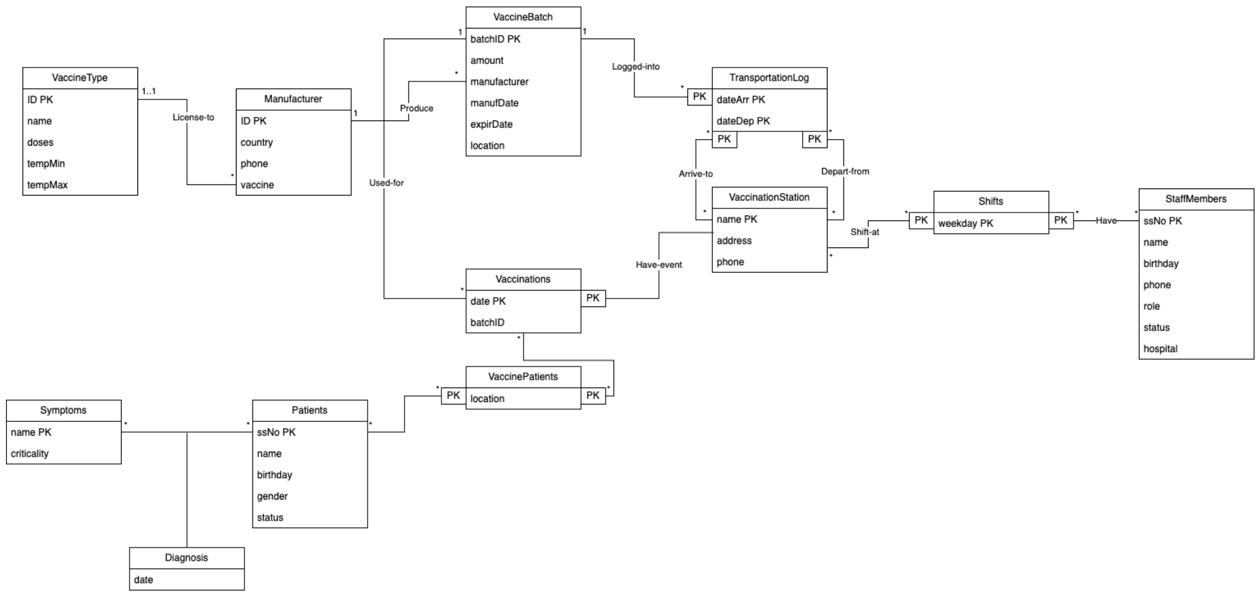
Are there any forms of redundancy or other anomalies in the database structure?

Since our relations in the database are in BCNF, most of the common anomalies are reduced, yet the relation Shifts(station, weekday, worker) and VaccinePatients(date, location, ssNo) contain redundancy, repeated information about station and weekday.

Part 2. Creating and analyzing the database

Changes from the first part

Given the feedback from the first part and the data, we have some changes to the UML of the database. The UML now is:



Creating and populating the database in Python

We used psycopg2, sqlalchemy, os and pandas to connect to the remote server, read the excel data and parse it. Pandas read the excel sheets into a dataframe, which we then split into tables. The tables were created and the queries were implemented in separate .sql files which we read and execute in the python script.

Data preparation

Before uploading the data to the database it needed to be cleaned and the inconsistent data had to be removed or fixed. First we matched the names of the columns in the dataframe created by pandas to the attributes in the SQL tables we created earlier. Most of the columns had the matching names already, but some had to be changed in the dataframe:

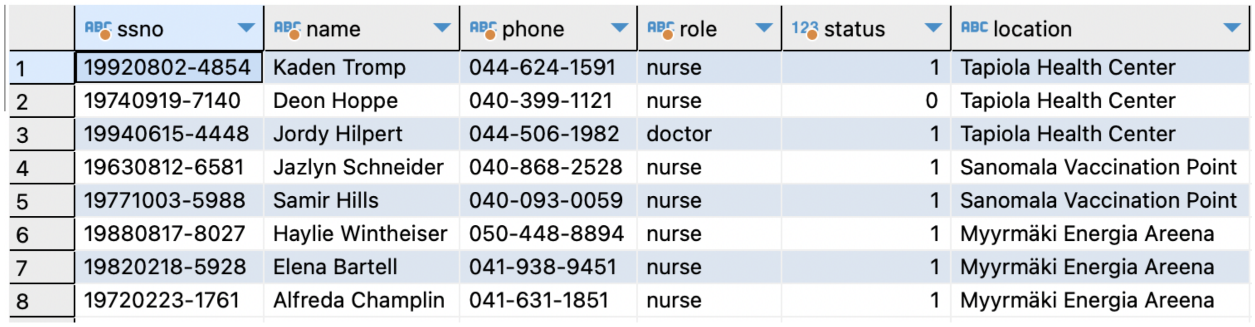
- in the ‘Transportation log’ sheet the name of the departure is ‘departure ‘ so it didn’t match with the ‘departure’ attribute and the last space was removed;
- the same problem was in the ‘Vaccinations’ sheet, the location column had an extra space in its name which had to be deleted;
- in the ‘Staffmembers’ and ‘Patients’ sheets some column names weren’t supported by the PostgreSQL syntax so the ‘social security number’, ‘date of birth’ and ‘vaccination status’ columns had to be renamed, too.

In the ‘Diagnosis’ sheet there were two tuples whose date was inconsistent. The first tuple, with the patient 730126-956K, had ‘2021-02-29’ as its date, although the last day of February in 2021 was 28th. The symptom reported by the patient wasn’t critical and it didn’t cause any further actions for the patient. This information isn’t crucial for the patient’s data, so we decided that it’s safe to assume the correct date. Our assumption is that the date was typed in wrong and the correct date is 2021-02-28. The date was then updated in the dataframe. The second tuple, with the patient 841229-112N, had its date in a format that couldn’t be recovered. The symptom wasn’t critical either, so we decided that it’s safe to delete the tuple from the dataframe.

Designing and running the queries

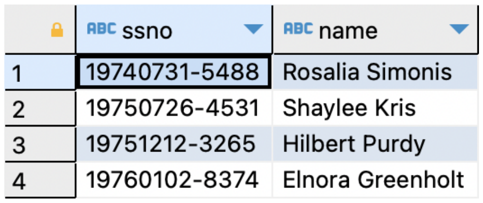
The results of the queries are taken from DBeaver for the sake of visual clarity. However, in the project the Python script executes a separate .sql file with the queries.

Query no. 1: Find all the staff members working in a vaccination on May 10, 2021. List the social security number, name, phone number, role, and the vaccination status of the staff member, as well as the location (name of the hospital or clinic) of the vaccination event.
Our query design: select the staff’s attributes (ssNo, name, phone, role, status) from the join of StaffMembers, Shifts and Vaccinations tables. May 10, 2021 is Monday, so we filter the staff based on this as well as the date.
Results:



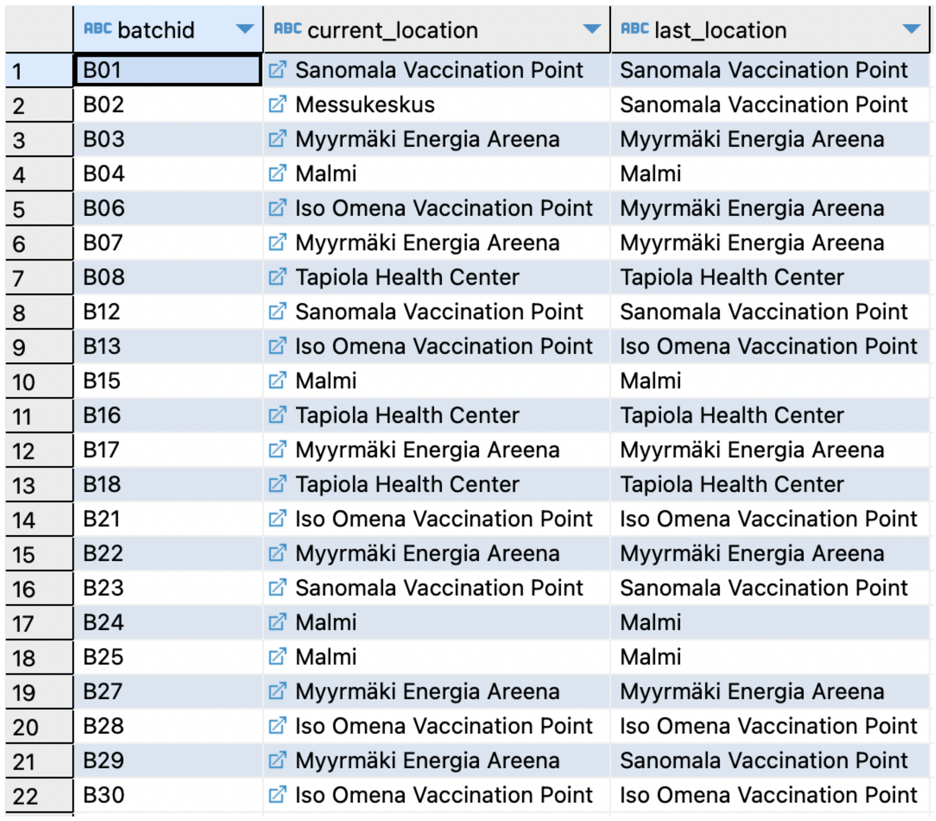
	ssno	name	phone	role	status	location
1	19920802-4854	Kaden Tromp	044-624-1591	nurse	1	Tapiola Health Center
2	19740919-7140	Deon Hoppe	040-399-1121	nurse	0	Tapiola Health Center
3	19940615-4448	Jordy Hilpert	044-506-1982	doctor	1	Tapiola Health Center
4	19630812-6581	Jazlyn Schneider	040-868-2528	nurse	1	Sanomala Vaccination Point
5	19771003-5988	Samir Hills	040-093-0059	nurse	1	Sanomala Vaccination Point
6	19880817-8027	Haylie Wintheiser	050-448-8894	nurse	1	Myyrmäki Energia Areena
7	19820218-5928	Elena Bartell	041-938-9451	nurse	1	Myyrmäki Energia Areena
8	19720223-1761	Alfreda Champlin	041-631-1851	nurse	1	Myyrmäki Energia Areena

Query no. 2: List all the doctors who would be available on Wednesdays in Helsinki.
Our query design: select the doctors’ names from the join of the StaffMembers, Shifts and VaccinationStations tables. The conditions for the tuples are ‘Wednesday’ as the weekday-attribute, ‘doctor’ as the role-attribute and address of the hospital containing ‘Helsinki’.
Results:



	ssno	name
1	19740731-5488	Rosalia Simonis
2	19750726-4531	Shaylee Kris
3	19751212-3265	Hilbert Purdy
4	19760102-8374	Elnora Greenholt

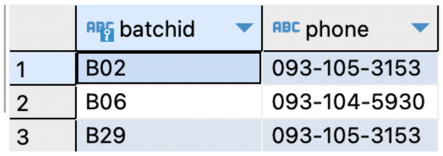
Query no. 3: For each vaccination batch, state the current location of the batch, and the last location in the transportation log.
Our interpretation: the location in the VaccineBatches-table describes the current location of the batches and is updated every time a batch is transported. However, there might be cases when it wasn't updated after a transportation and is not up to date. In that case the correct current location of the batch is the latest arrival hospital and is checked from the transportation log.
Our query design: first a subquery selects batchIDs and the latest date for each batchID in the Transportation log, then the resulting relation is joined with VaccineBatch and Transportation log to produce the current location of each batchID in the VaccineBatch and the corresponding latest arrival hospital in the log.
Results:



	batchid	current_location	last_location
1	B01	Sanomala Vaccination Point	Sanomala Vaccination Point
2	B02	Messukeskus	Sanomala Vaccination Point
3	B03	Myyrmäki Energia Areena	Myyrmäki Energia Areena
4	B04	Malmi	Malmi
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	B12	Sanomala Vaccination Point	Sanomala Vaccination Point
9	B13	Iso Omena Vaccination Point	Iso Omena Vaccination Point
10	B15	Malmi	Malmi
11	B16	Tapiola Health Center	Tapiola Health Center
12	B17	Myyrmäki Energia Areena	Myyrmäki Energia Areena
13	B18	Tapiola Health Center	Tapiola Health Center
14	B21	Iso Omena Vaccination Point	Iso Omena Vaccination Point
15	B22	Myyrmäki Energia Areena	Myyrmäki Energia Areena
16	B23	Sanomala Vaccination Point	Sanomala Vaccination Point
17	B24	Malmi	Malmi
18	B25	Malmi	Malmi
19	B27	Myyrmäki Energia Areena	Myyrmäki Energia Areena
20	B28	Iso Omena Vaccination Point	Iso Omena Vaccination Point
21	B29	Myyrmäki Energia Areena	Sanomala Vaccination Point
22	B30	Iso Omena Vaccination Point	Iso Omena Vaccination Point

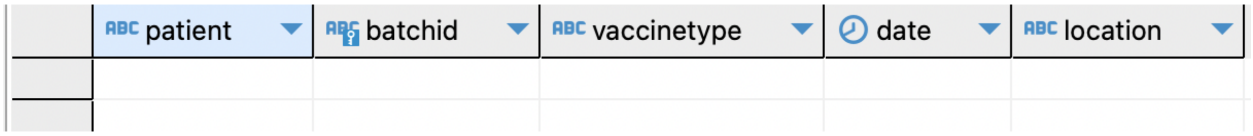
List separately the vaccine batch numbers with inconsistent location data, along with the phone number of the hospital or clinic where the vaccine batch should currently be.

Our interpretation: the hospitals where batches should be are the latest arrival hospitals in the transportation log.
Out query design: similar to the previous query, except that this time we compare the location of each batchID in the VaccineBatch to the latest arrival hospital in the log, and select only the tuples where these locations don't match. We also select the phone numbers of the latest arrival hospitals for these batches.
Results:



	batchid	phone
1	B02	093-105-3153
2	B06	093-104-5930
3	B29	093-105-3153

Query no. 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.
Our query design: selecting the required attributes from the join of tables VaccineBatch, Diagnosis, VaccineType, Vaccinations, Symptoms, VaccinePatients and Manufacturer. The conditions for the tuples are: 1 as the criticality-attribute and date>'2021-05-10' for the Diagnosis' columns.
Results: empty list. After running a query to check which patients reported critical symptoms we see that there's only four of them, and none of them reported the symptoms after May 10, 2021. Thus the query which filters the reported symptoms by date returns an empty list.

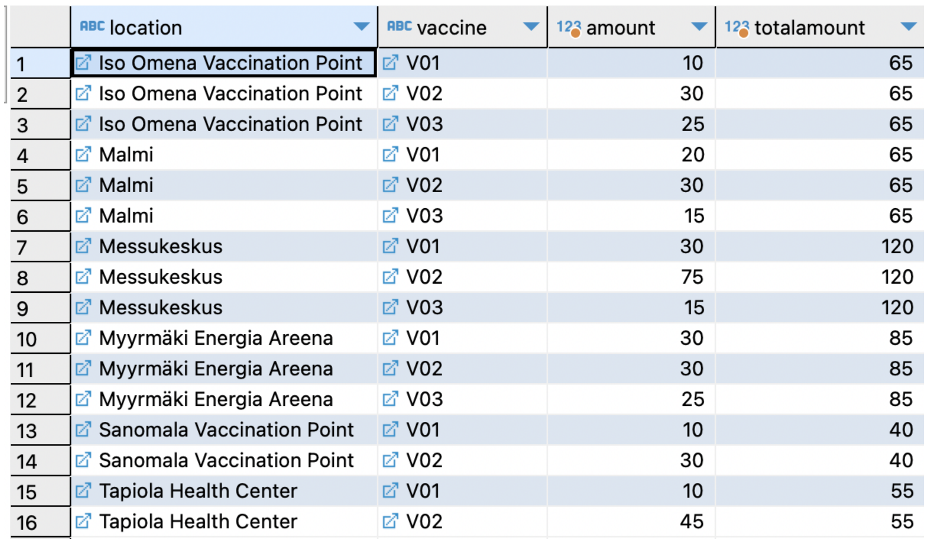


	patient	batchid	vaccinetype	date	location

Query no. 5: Create a view for patients with an additional column "vaccinationStatus". This column takes the value 1 if the patient has attended enough vaccinations, and 0 otherwise.
Our interpretation: patients with two or more doses of vaccine(s) of any type(s) have a vaccination status 1 (two doses of different types included in this case), otherwise it's 0.
Our query design: first a subquery counts how many times each patient's ssNo appears in the VaccinePatients table. Then a view is created and the attribute 'vaccinationstatus' gets a value 1 if the subquery returns >=2 for a ssNo and 0 otherwise.
Results: The result is given in the "query_5_result.csv"

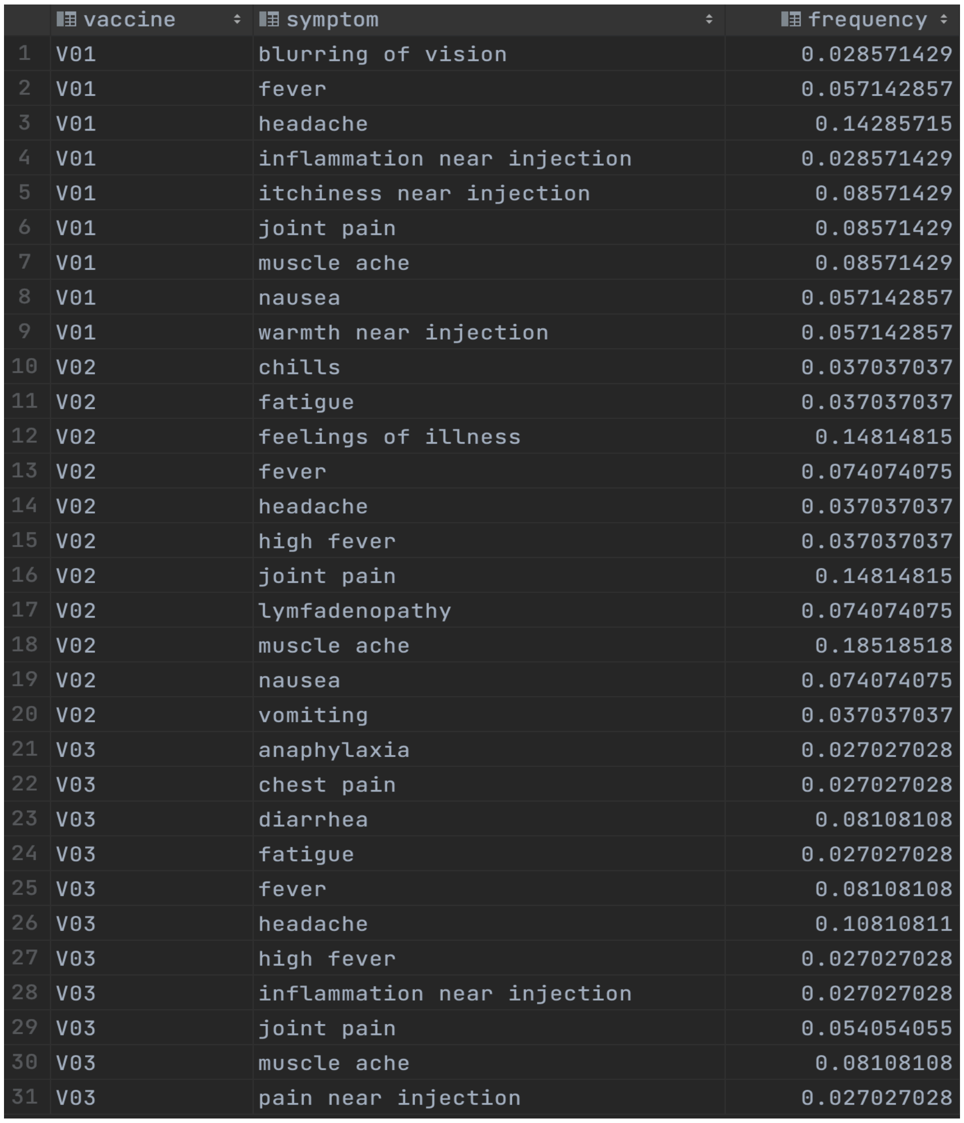
Query no. 6: Find the total number of vaccines stored in each hospital and clinic. You should consider only those vaccine batches in your database that are located in the hospital. For each hospital or clinic you should list both the total number of vaccines and number of vaccines of different types.

Our interpretation: only the locations in the VaccineBatch-table need to be considered.
Our query design: first a subquery selects locations and counts amounts of doses of each vaccine type from the VaccineBatch-table and groups them by hospital. Then an outer query counts the total amount of vaccines in every hospital from the VaccineBatch-table and groups them by hospital, too.
Results:



	location	vaccine	amount	totalamount
1	Iso Omena Vaccination Point	V01	10	65
2	Iso Omena Vaccination Point	V02	30	65
3	Iso Omena Vaccination Point	V03	25	65
4	Malmi	V01	20	65
5	Malmi	V02	30	65
6	Malmi	V03	15	65
7	Messukeskus	V01	30	120
8	Messukeskus	V02	75	120
9	Messukeskus	V03	15	120
10	Myyrmäki Energia Areena	V01	30	85
11	Myyrmäki Energia Areena	V02	30	85
12	Myyrmäki Energia Areena	V03	25	85
13	Sanomala Vaccination Point	V01	10	40
14	Sanomala Vaccination Point	V02	30	40
15	Tapiola Health Center	V01	10	55
16	Tapiola Health Center	V02	45	55

Query no. 7: For each vaccine type, you should find the average frequency of different symptoms diagnosed. The symptom should not be considered to be caused by the vaccine, if it has been diagnosed before the patient got the vaccine. If a patient has received two different types of vaccines before the diagnosis of the symptom, the symptom should be counted once for both of the vaccines.
Our interpretation: The table diagnosis contains the records of patients' symptoms but some are not caused by vaccinations. The symptom must have the recorded date on the same or after the vaccination date to be considered as caused by the vaccine.
Our query design: First, find the patients with the recorded symptom(s) caused by the first and second doses and the vaccine type that caused that symptom(s). Then count the number of cases reported for different symptoms and vaccines. Second, find the total number of patients using a given vaccine. Then the result is the division of the number of cases and the total number of patients.
Results: (on the next page)



	vaccine	symptom	frequency
1	V01	blurring of vision	0.028571429
2	V01	fever	0.057142857
3	V01	headache	0.14285715
4	V01	inflammation near injection	0.028571429
5	V01	itchiness near injection	0.08571429
6	V01	joint pain	0.08571429
7	V01	muscle ache	0.08571429
8	V01	nausea	0.057142857
9	V01	warmth near injection	0.057142857
10	V02	chills	0.037037037
11	V02	fatigue	0.037037037
12	V02	feelings of illness	0.14814815
13	V02	fever	0.074074075
14	V02	headache	0.037037037
15	V02	high fever	0.037037037
16	V02	joint pain	0.14814815
17	V02	lymfadenopathy	0.074074075
18	V02	muscle ache	0.18518518
19	V02	nausea	0.074074075
20	V02	vomiting	0.037037037
21	V03	anaphylaxia	0.027027028
22	V03	chest pain	0.027027028
23	V03	diarrhea	0.08108108
24	V03	fatigue	0.027027028
25	V03	fever	0.08108108
26	V03	headache	0.10810811
27	V03	high fever	0.027027028
28	V03	inflammation near injection	0.027027028
29	V03	joint pain	0.054054055
30	V03	muscle ache	0.08108108
31	V03	pain near injection	0.027027028

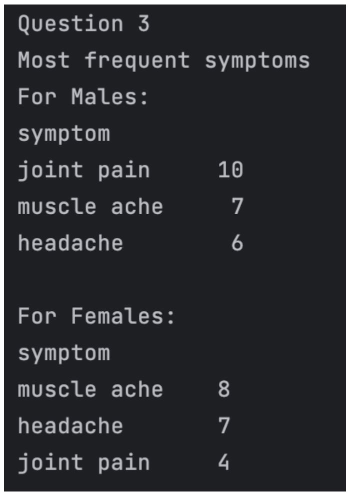
Part 3. Data analysis

Query no. 1: Create a dataframe for patients and symptoms containing the following columns: (1) ssNO, (2) gender, (3) dateOfBirth, (4) symptom, (5) diagnosisDate. Create a table named ”PatientSymptoms” using the command to sql with options index = True, if exists = "replace".
Our design: we read a query and write its results into a dataframe, which we then write to the database using to_sql() and sqlalchemy.commit().
Results: in the query1.xlsx file.

Query no. 2: Create a dataframe for patients and vaccines containing the following columns: (1) patientssNO, (2) date1, (3) vaccinetype1, (4) date2, (5) vaccinetype2. The attribute ”date1” and ”date2” refer to the date when the first and/or second dose were given to a patient respectively. Similarly, ”vaccinetype1” and ”vaccinetype2” are the type of vaccine used for the first and/or

second dose. The value of the attribute should be NULL if the patient has not received some dose. Create a table named ”PatientVaccineInfo” using the dataframe as in Task 1.
Our design: same as for query no. 1
Results: in the query2.xlsx file.

Query no. 3: Create a dataframe using the table ”PatientSymptoms” and separate it into two dataframes, one for males and one for females. What are the top three most common symptoms for males and females?
Our design: we read a query and write its results into a dataframe, which we then split into tw dataframes based on gender, and then we count how many times each symptom appears in the tables.
Results:



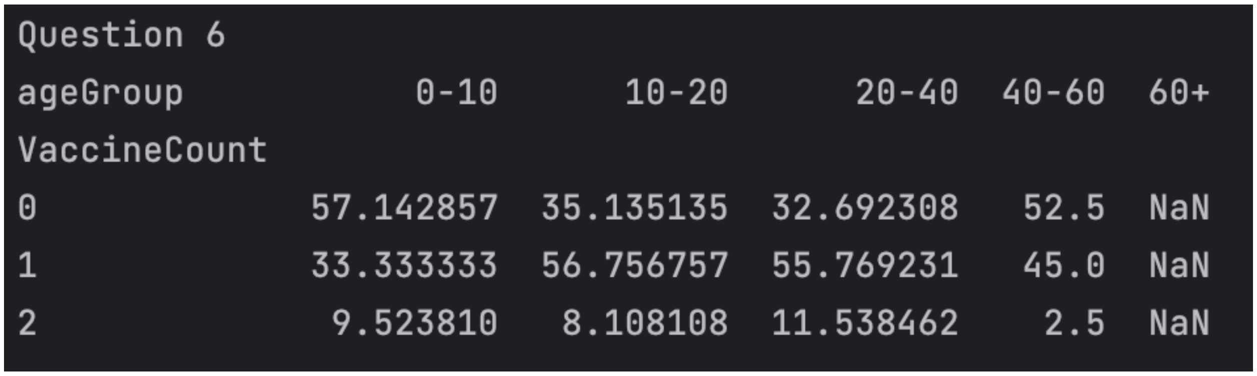
```
Question 3
Most frequent symptoms
For Males:
symptom
joint pain     10
muscle ache     7
headache        6

For Females:
symptom
muscle ache    8
headache       7
joint pain     4
```

Query no. 4: Create a dataframe using table ”Patient” and add the ”ageGroup” column for each patient. The age groups are ”0-10”, ”10-20”, ”20-40”, ”40-60”, ”60+”
Out interpretation: The age groups are [0,9], [10,19], [20,39], [40,59], [60, inf]. That is, a person who is 39 years old will be in the ‘20-39’ group and a 40-year-old person will be in the ‘40-59’ group.
Our design: we read a query and write its results into a dataframe. We then use pd.cut() to “slice” it according to bins and
Results: in the query4.xlsx file.

Query no. 5: Using the same dataframe as in the previous step, add a column describing each patient’s vaccination status. The statuses are defined as ”0” for not vaccinated, ”1” for vaccinated once, and ”2” for fully-vaccinated.
Our design: we count the not null values of date1 and date2 and use them to make a new column with the vaccine count.
Results: in the query5.xlsx file.

Query no. 6: For each age group, calculate the percentage of people who have received zero, one, or two doses of vaccines. Show the results in a dataframe, where the index is the

vaccination status from task (5) and the columns are the age groups. The sum over each age group column should be 100%. EXTRA: Solve this task using pivoting.
<u>Our design</u>: Using the df from the 5th question, we used pandas' pivot_table to create a new df with the required data. A lambda function for calculating the percentages is then applied to the df.
<u>Results</u>:



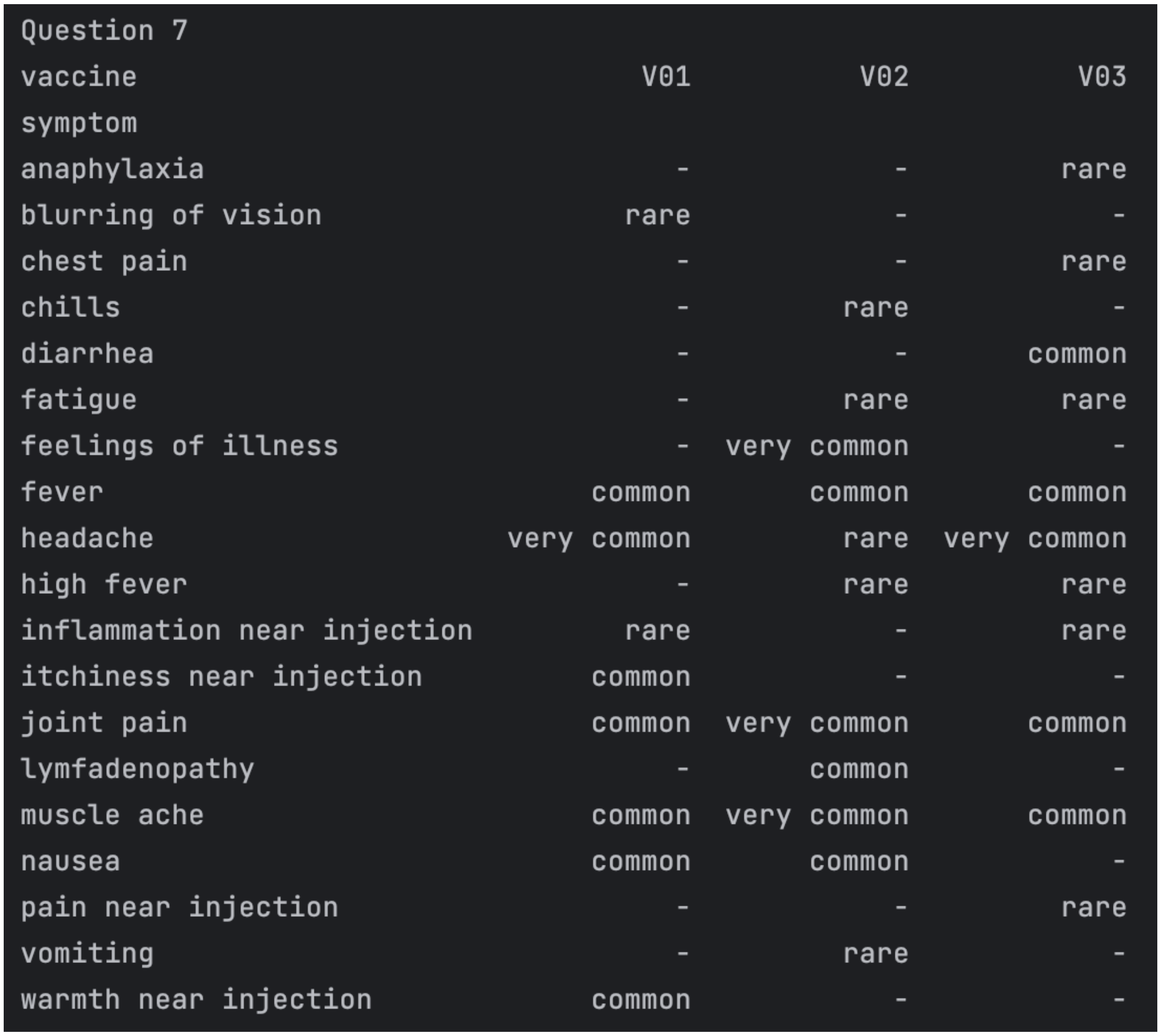
```
Question 6
ageGroup            0-10       10-20       20-40  40-60  60+
VaccineCount
0              57.142857  35.135135  32.692308   52.5  NaN
1              33.333333  56.756757  55.769231   45.0  NaN
2               9.523810   8.108108  11.538462    2.5  NaN
```

As we can see, no one over 60 y.o have received any vaccines. By checking the patients' data we can see that the oldest persons are under 60 y.o so the results of the query make sense.

Query no. 7: Create a dataframe for symptoms with three additional columns: 'V01', 'V02', and 'V03'. The columns should tell the relative frequency of the symptom with the following values:

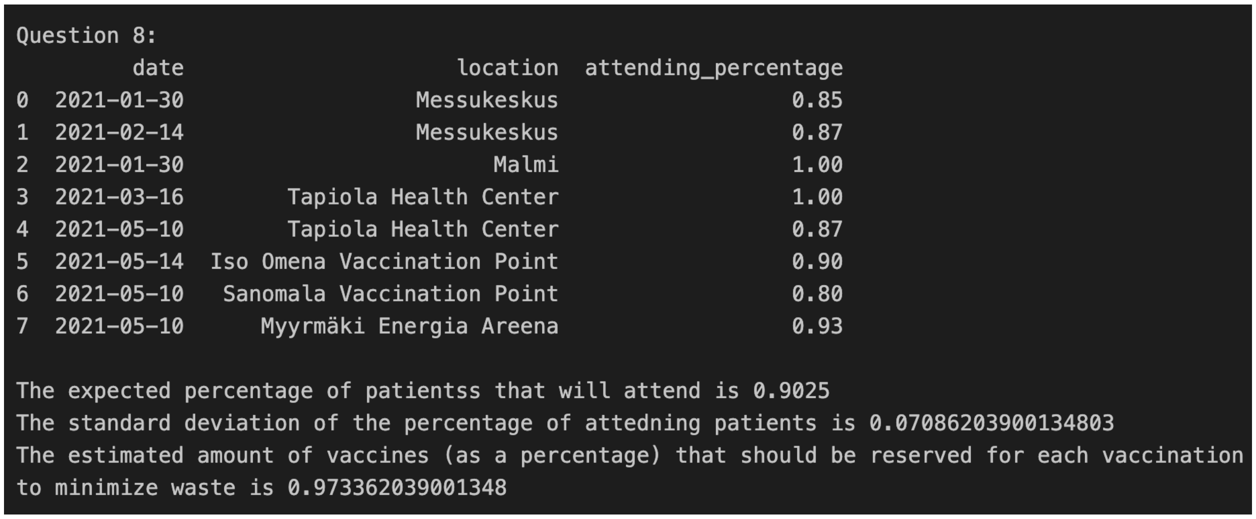
≥ 0.1	"very common"
≥ 0.05	"common"
> 0.0	"rare"
0.0	"-"

<u>Our design</u>: first we use the same query that we used in the 7th question in part II of the project. It gives us the frequencies of each symptom for different vaccine types, performing a causality check for each symptom. Then we pivot the resulting df using pivot_table. Then a function which labels the frequencies is applied.
<u>Results</u>: (on the next page)



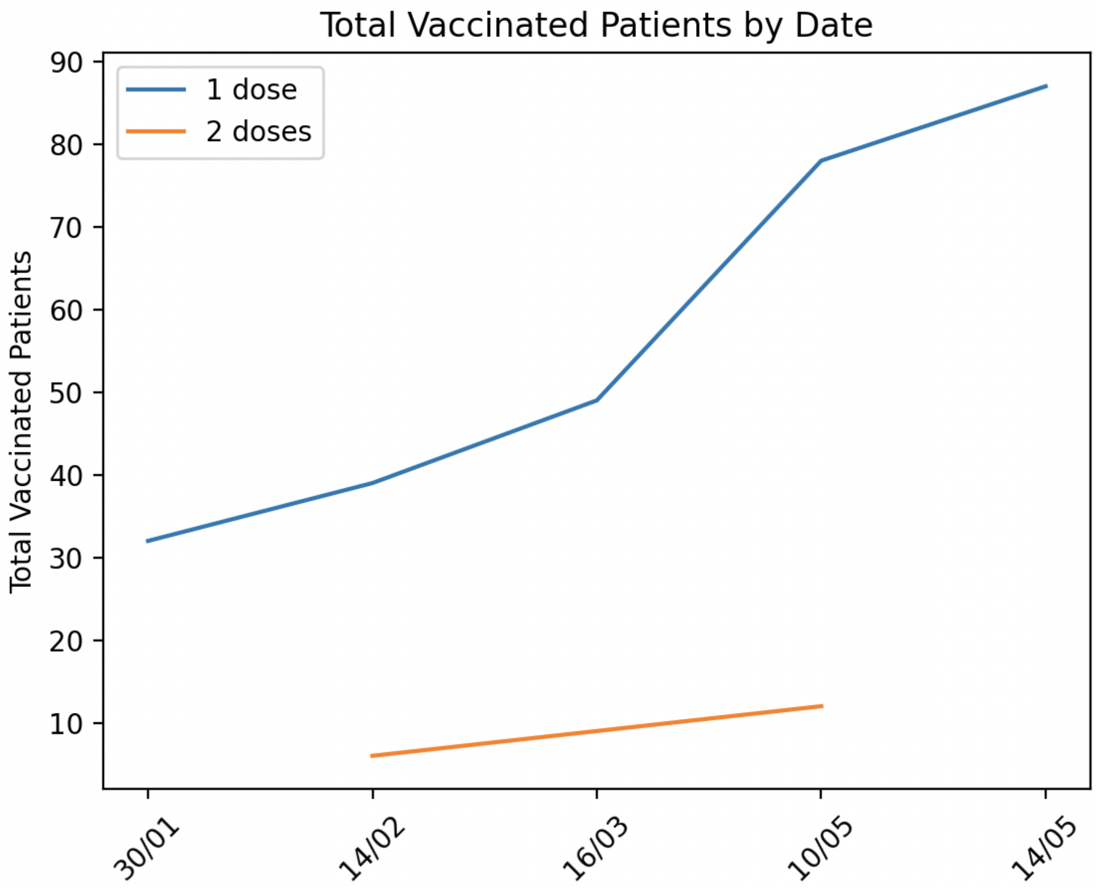
```
Question 7
vaccine                               V01          V02          V03
symptom
anaphylaxia                             -            -         rare
blurring of vision                   rare            -            -
chest pain                              -            -         rare
chills                                  -         rare            -
diarrhea                                -            -       common
fatigue                                 -         rare         rare
feelings of illness                     -  very common            -
fever                              common       common       common
headache                      very common         rare  very common
high fever                              -         rare         rare
inflammation near injection          rare            -         rare
itchiness near injection           common            -            -
joint pain                         common  very common       common
lymfadenopathy                          -       common            -
muscle ache                        common  very common       common
nausea                             common       common            -
pain near injection                     -            -         rare
vomiting                                -         rare            -
warmth near injection              common            -            -
```

Query no. 8: Estimate the amount of vaccines (as a percentage) that should be reserved for each vaccination to minimize waste. Do this by first finding the expected percentage of patients that will attend and increase the number by Standard Deviation (STD) of the percentage of attending patients.
Our design: First, we calculate the percentages of attending patients from past vaccination events by taking the number of attending patients divided by the amount of vaccines used in each event. The result is the expected (mean) percentage plus the standard deviation of the percentage.
Results: (on the next page)



```
Question 8:
        date                    location  attending_percentage
0  2021-01-30                Messukeskus                  0.85
1  2021-02-14                Messukeskus                  0.87
2  2021-01-30                      Malmi                  1.00
3  2021-03-16       Tapiola Health Center                 1.00
4  2021-05-10       Tapiola Health Center                 0.87
5  2021-05-14  Iso Omena Vaccination Point                0.90
6  2021-05-10   Sanomala Vaccination Point                0.80
7  2021-05-10      Myyrmäki Energia Areena                0.93

The expected percentage of patientss that will attend is 0.9025
The standard deviation of the percentage of attedning patients is 0.07086203900134803
The estimated amount of vaccines (as a percentage) that should be reserved for each vaccination
to minimize waste is 0.973362039001348
```

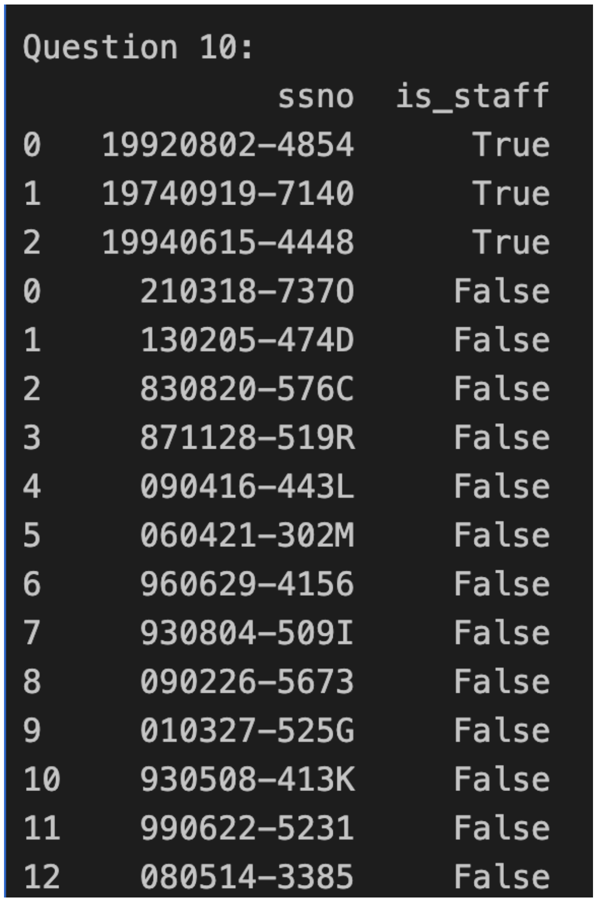
Query no. 9: Plot the total number of vaccinated patients with respect to date (Hint: functions cumsum() and strftime()). EXTRA: Plot the number of patients who have gotten two doses to the same figure.
Our design: first we use two separate queries to find all the patients who have received one dose of vaccine (date1 != null) and two doses of vaccine (date2 != null) and store the data in two dataframes. Then we convert the dates to a datetime format using the to_datetime() function and change the output format with the strftime() function. We calculate the cumulative count of vaccinated patients using the cumsum() function and then print the two lines to the same plot.
Results: (on the next page)



Query no. 10: Suppose that we found out that the nurse with ssNo ”19740919-7140” has been tested positive for corona on 15.5.2021. You should find the social security numbers and names of the patients and staff members that the nurse may have met in vaccination events in the past 10 days? (You are allowed to solve this task using multiple steps and queries).

Our design: We decided to generalize the problem by creating a function that takes two arguments, of which the first one is the social security of the staff tested positive and the second is the date he/she received the test result. The result is a dataframe containing the social security numbers of the staff and the patients along with a column containing boolean values indicating whether the person is a staff or not.

Result: (On the next page)



```
Question 10:
              ssno  is_staff
0   19920802-4854      True
1   19740919-7140      True
2   19940615-4448      True
0     210318-737O     False
1     130205-474D     False
2     830820-576C     False
3     871128-519R     False
4     090416-443L     False
5     060421-302M     False
6     960629-4156     False
7     930804-509I     False
8     090226-5673     False
9     010327-525G     False
10    930508-413K     False
11    990622-5231     False
12    080514-3385     False
```