МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

|  |
| --- |
| КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ |

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Канд. техн. наук, доцент |  |  |  | Е.Л.Турнецкая |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| Управление данными в графической базе данных NEO4J |
| по дисциплине: Методология и технология проектирования информационных систем |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | Z4410МК |  | 12.12.24 |  | Е.Ю. Чахунашвили |
|  | номер группы |  | подпись, дата |  | инициалы, фамилия |
| Студенческий билет № | 2024/4009 | |  |  |  |

Санкт-Петербург 2024

**Цель работы**

Получение практических навыков работы с графической базой данных Neo4j.

**Задание**

1. Установить Docker.

2. Создать контейнер для развертывания Neo4j.

3. Выполнить упражнения по работе с БД Sales.

4. По своему варианту, который равен номеру по списку группы спроектировать и реализовать графовую БД (варианты вынесены в отдельный файл).

5. Заполнить ее 7-10 узлами для каждой метки (7-10 записей в каждой таблице, если мы переходим на понятия реляционной БД).

6. Продумать 10 запросов разной степени сложности и зафиксировать их в письменной форме.

7. Реализовать запросы на Chyper в клиентской части БД (примеры показаны в упражнениях).

8. Выполнить отчет. В отчете должен быть вывод в формате эссе.

**Вариант 11. Поликлиника**

В Хоббитонии есть центральная поликлиника «Имени Дж. Р. Р. Толкина» для граждан страны. В поликлинике регистрируются случаи заболевания разными болезнями (таблица Учет посещений пациентов) и делаются прививки (таблица с учетом прививок). Прививки могут дать иммунитет от некоторых болезней. После некоторых болезней так же вырабатывается иммунитет. Некоторыми болезнями (например, простуда), хоббиты могут болеть несколько раз. При каждом обращении регистрируется дата начала заболевания и дата выздоровления (если удалось спасти, конечно). Главный санитарный врач Хоббитонии – Б. Бэггинс обращается к Вам для разработки базы по учету здоровья нации и предотвращения эпидемий.

**Результат выполнения**

Для начала работы с Neo4j создали контейнер с именем policlinic в Docker, воспользовавшись Bash-командой:



Созданный контейнер в Docker представлен на рисунке 1.

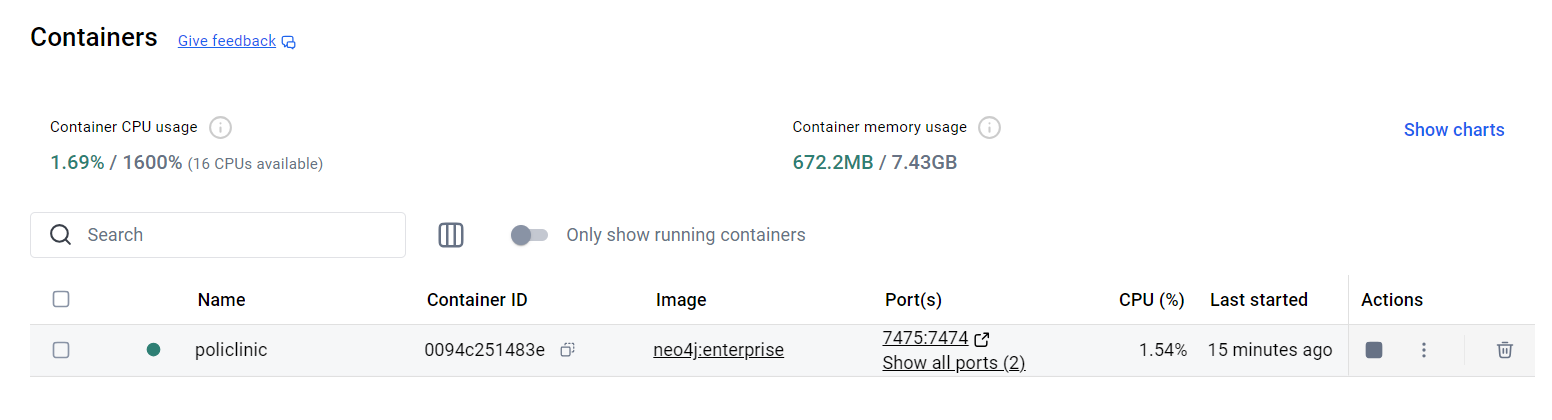


Рисунок 1 – созданный контейнер в Docker

Созданная база данных policnic представлена на рисунке 2.



Рисунок 2 – созданная база данных в Neo4j

Для заполнения БД данными было необходимо продумать структуру хранения данных, а именно разработать реляционную БД. База данных состоит из пяти таблиц:

* Patient: в таблице хранятся данные о пациентах, которые зарегистрированы в поликлинике (имя, фамилия, дата рождения).
* Illness: в таблице хранятся данные о заболеваниях, которыми могут болеть пациенты (код и название заболевания).
* Vaccination: в таблице хранятся данные о вакцинах, которые могут сделать пациентам (код и название вакцины).  
  PatientVisits: в таблице хранятся данные о визитах пациентов в поликлинику, а также даты начала заболевания и даты выздоровления.  
  AccountingVaccination: в таблице хранятся данные о пациентах и вакцинах, которые они сделали в поликлинике.

На рисунке 3 представлена реляционная база данных.

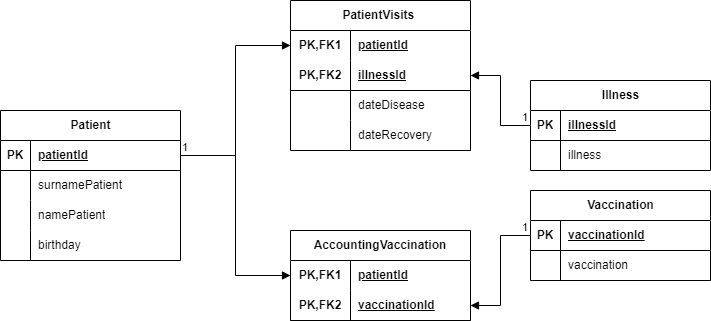


Рисунок 3 – реляционная база данных

SQL-код создания таблиц в реляционной базе данных представлен ниже:

CREATE TABLE Patient(

patientId INT IDENTITY(1,1) PRIMARY KEY,

surnamePatient VARCHAR(50) not null,

namePatient VARCHAR(50) not null,

birthday DATE not null

);

CREATE TABLE Vaccination(

vaccinationId INT IDENTITY(1,1) PRIMARY KEY,

vaccination VARCHAR(100) not null

);

CREATE TABLE Illness(

illnessId INT IDENTITY(1,1) PRIMARY KEY,

illness VARCHAR(100) not null

);

CREATE TABLE AccountingVaccination(

patientId INT,

vaccinationId INT,

CONSTRAINT FK\_AccountingVaccination\_To\_Patient FOREIGN KEY (patientId) REFERENCES Patient(patientId),

CONSTRAINT FK\_AccountingVaccination\_To\_Vaccination FOREIGN KEY (vaccinationId) REFERENCES Vaccination(vaccinationId)

);

CREATE TABLE PatientVisits (

patientId INT,

illnessId INT,

dateDisease DATE not null,

dateRecovery DATE not null,

CONSTRAINT FK\_PatientVisits\_To\_Patient FOREIGN KEY (patientId) REFERENCES Patient(patientId),

CONSTRAINT FK\_PatientVisits\_To\_Illness FOREIGN KEY (illnessId) REFERENCES Illness(illnessId)

);

SQL-код заполнения таблиц базы данных поликлиники представлен ниже:

INSERT INTO Patient (surnamePatient, namePatient, birthday)

VALUES

('Громова', 'Виктория', '1998-05-15'),

('Чернышева', 'София', '1996-10-03'),

('Дроздов', 'Антон', '2005-06-17'),

('Богомолов', 'Илья', '1980-11-10'),

('Самойлова', 'Татьяна', '2000-12-12'),

('Щербакова', 'Анастасия', '1996-01-12'),

('Пугачев', 'Даниил', '2008-04-16'),

('Панов', 'Юрий', '1992-09-23'),

('Блохина', 'Ксения', '2001-10-08'),

('Зубков', 'Степан', '1997-06-14')

SELECT \* FROM Patient;

INSERT INTO Illness(illness)

VALUES

('Бронхит'),

('Аллергия'),

('Коклюш'),

('Герпес'),

('Пневмония'),

('Корь'),

('Астма'),

('Гепатит B'),

('COVID-19'),

('Менингит')

SELECT \* FROM Illness;

INSERT INTO Vaccination(vaccination)

VALUES

('Холера'),

('Дифтерия'),

('Коклюш'),

('Бешенство'),

('Пневмония'),

('Корь'),

('Паротит'),

('Гепатит B'),

('COVID-19'),

('Менингит')

SELECT \* FROM Vaccination;

INSERT INTO PatientVisits(patientId, illnessId, dateDisease, dateRecovery)

VALUES

(1, 1,'2024-10-15', '2024-10-25'),

(4, 2,'2024-03-05', '2024-03-06'),

(3, 9,'2024-04-03', '2024-04-14'),

(6, 7,'2024-05-19', '2024-06-01'),

(4, 2,'2024-07-08', '2024-07-09'),

(6, 1,'2024-10-02', '2024-10-24'),

(7, 6,'2024-11-15', '2024-11-20')

SELECT \* FROM PatientVisits;

INSERT INTO AccountingVaccination(patientId, vaccinationId)

VALUES

(3, 9),

(7, 6),

(4, 4),

(5, 8),

(8, 10),

(9, 1),

(2, 9),

(10, 9)

SELECT \* FROM AccountingVaccination;

Работа с графовой базой данных подразумевает собой определение узлов и ребер. Так, сущности Patient, Illness и Vaccination будут узлами, а сущности PatientVisits и AccountingVaccination — ребрами между узлами.

Для заполнения БД поликлиники использовали Chyper-код, который представлен ниже:

CREATE

(patient1:Patient {patientId:1, surnamePatient: 'Громова', namePatient:'Виктория', birthday:date('1998-05-15')}),

(patient2:Patient {patientId:2, surnamePatient:'Чернышева', namePatient:'София', birthday:date('1996-10-03')}),

(patient3:Patient {patientId:3, surnamePatient: 'Дроздов', namePatient:'Антон', birthday:date('2005-06-17')}),

(patient4:Patient {patientId:4, surnamePatient:'Богомолов', namePatient:'Илья', birthday:date('1980-11-10')}),

(patient5:Patient {patientId:5, surnamePatient:'Самойлова', namePatient:'Татьяна', birthday:date('2000-12-12')}),

(patient6:Patient {patientId:6, surnamePatient: 'Щербакова', namePatient:'Анастасия', birthday:date('1996-01-12')}),

(patient7:Patient {patientId:7, surnamePatient:'Пугачев', namePatient:'Даниил', birthday:date('2008-04-16')}),

(patient8:Patient {patientId:8, surnamePatient: 'Панов', namePatient:'Юрий', birthday:date('1992-09-23')}),

(patient9:Patient {patientId:9, surnamePatient: 'Блохина', namePatient:'Ксения', birthday:date('2001-10-08')}),

(patient10:Patient {patientId:10, surnamePatient:'Зубков', namePatient:'Степан', birthday:date('1997-06-14')})

CREATE

(vaccination1: Vaccination {vaccinationId: 1, vaccination: 'Холера'}),

(vaccination2: Vaccination {vaccinationId: 2, vaccination: 'Дифтерия'}),

(vaccination3: Vaccination {vaccinationId: 3, vaccination: 'Коклюш'}),

(vaccination4: Vaccination {vaccinationId: 4, vaccination: 'Бешенство'}),

(vaccination5: Vaccination {vaccinationId: 5, vaccination: 'Пневмония'}),

(vaccination6: Vaccination {vaccinationId: 6, vaccination: 'Корь'}),

(vaccination7: Vaccination {vaccinationId: 7, vaccination: 'Паротит'}),

(vaccination8: Vaccination {vaccinationId: 8, vaccination: 'Гепатит B'}),

(vaccination9: Vaccination {vaccinationId: 9, vaccination: 'COVID-19'}),

(vaccination10: Vaccination {vaccinationId: 10, vaccination: 'Менингит'})

CREATE

(illness1: Illness {illnessId: 1, illness:'Бронхит' }),

(illness2: Illness {illnessId: 2, illness: 'Аллергия'}),

(illness3: Illness {illnessId: 3, illness: 'Коклюш'}),

(illness4: Illness {illnessId: 4, illness: 'Герпес'}),

(illness5: Illness {illnessId: 5, illness:'Пневмония'}),

(illness6: Illness {illnessId: 6, illness: 'Корь'}),

(illness7: Illness {illnessId: 7, illness: 'Астма'}),

(illness8: Illness {illnessId: 8, illness: 'Гепатит B'}),

(illness9: Illness {illnessId: 9, illness: 'COVID-19'}),

(illness10: Illness {illnessId: 10, illness: 'Менингит'})

CREATE

(patient1)-[:PatientVisits {dateDisease:date('2024-10-15'), dateRecovery:date('2024-10-25')}]->(illness1),

(patient4)-[:PatientVisits {dateDisease:date('2024-03-05'), dateRecovery:date('2024-03-06')}]->(illness2),

(patient3)-[:PatientVisits {dateDisease:date('2024-04-03'), dateRecovery:date('2024-04-14')}]->(illness9),

(patient6)-[:PatientVisits {dateDisease:date('2024-05-19'), dateRecovery:date('2024-06-01')}]->(illness7),

(patient4)-[:PatientVisits {dateDisease:date('2024-07-08'), dateRecovery:date('2024-07-09')}]->(illness2),

(patient6)-[:PatientVisits {dateDisease:date('2024-10-02'), dateRecovery:date('2024-10-24')}]->(illness1),

(patient7)-[:PatientVisits {dateDisease:date('2024-11-15'), dateRecovery:date('2024-11-20')}]->(illness6)

CREATE

(patient3)-[:AccountingVaccination]->(vaccination9),

(patient7)-[:AccountingVaccination]->(vaccination6),

(patient4)-[:AccountingVaccination]->(vaccination4),

(patient5)-[:AccountingVaccination]->(vaccination8),

(patient8)-[:AccountingVaccination]->(vaccination9),

(patient9)-[:AccountingVaccination]->(vaccination1),

(patient2)-[:AccountingVaccination]->(vaccination9),

(patient10)-[:AccountingVaccination]->(vaccination9),

(patient8)-[:AccountingVaccination]->(vaccination9)

В результате, получили созданные с помощью кода узлы и связи между ними. На рисунке 4 представлен графический вид БД.

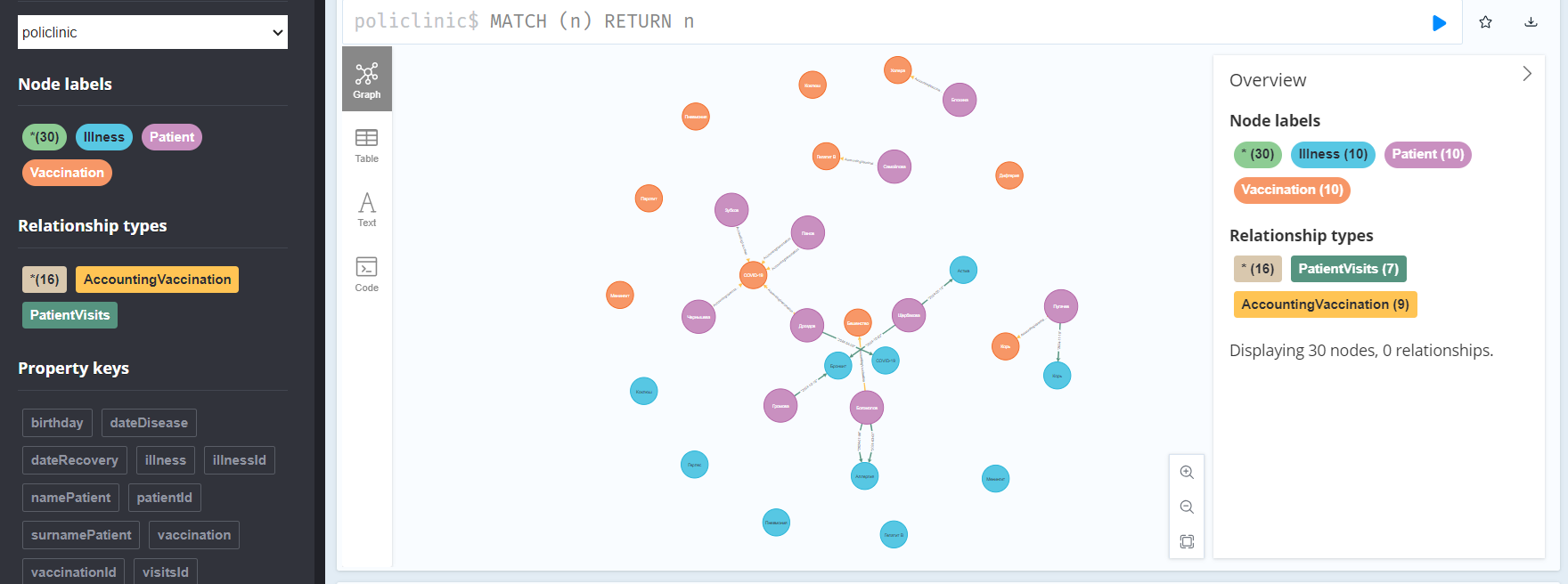


Рисунок 4 – графический вид БД в Neo4j

Далее, будут представлены запросы разной степени сложности для базы данных поликлиники и результаты этих запросов.

1. Вывести всех пациентов, заболевших аллергией.

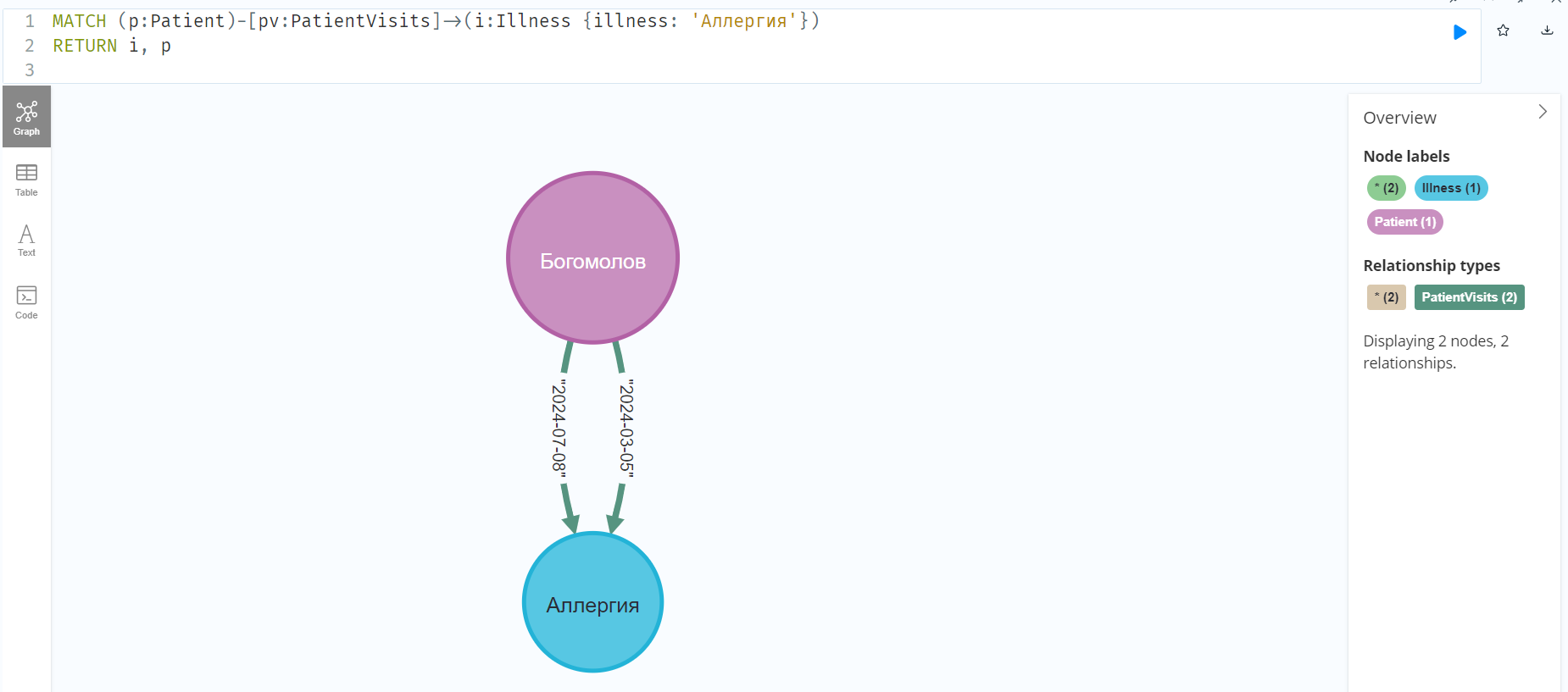


Рисунок 5 – Список пациентов, заболевших аллергией

1. Вывести список всех вакцин, которые можно сделать в поликлинике.

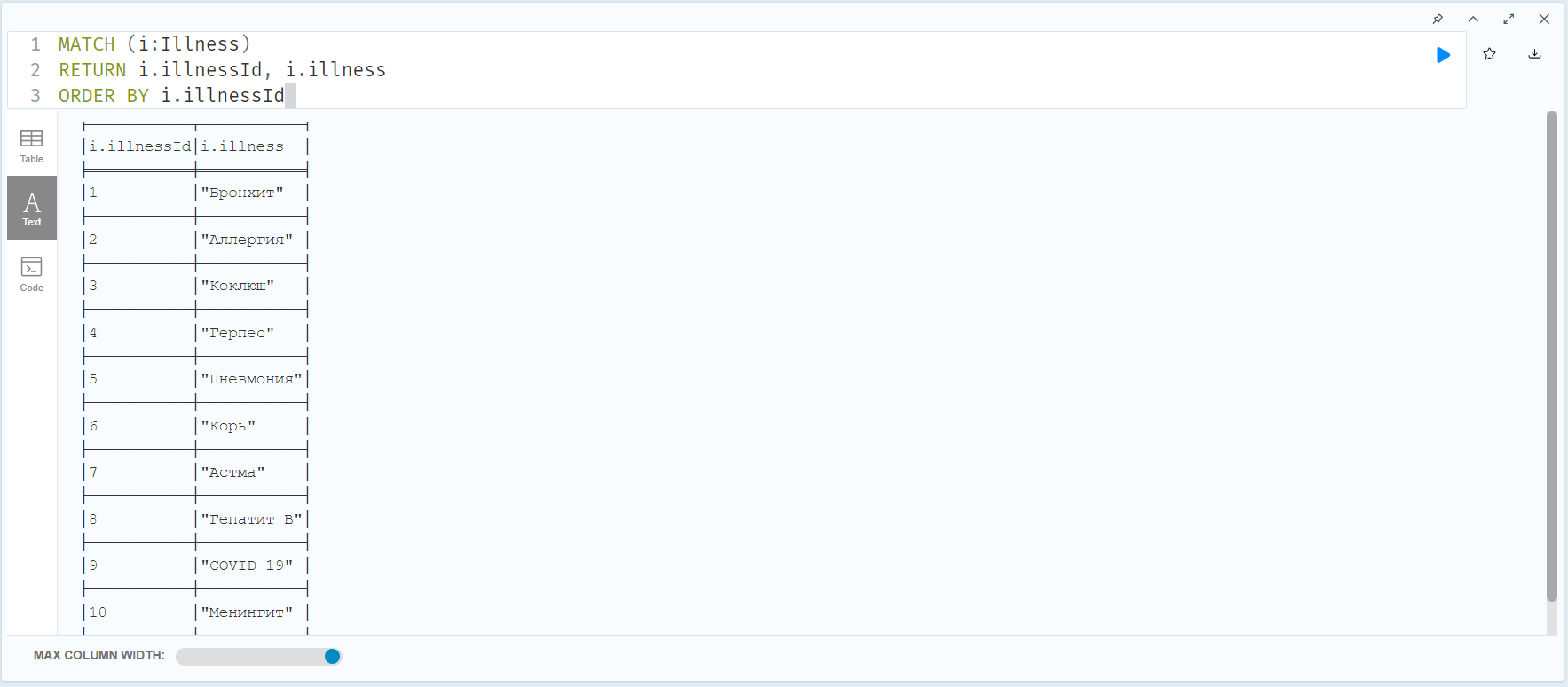


Рисунок 6 – Список вакцин поликлиники

1. Вывести список всех вакцин, сделанных в поликлинике.

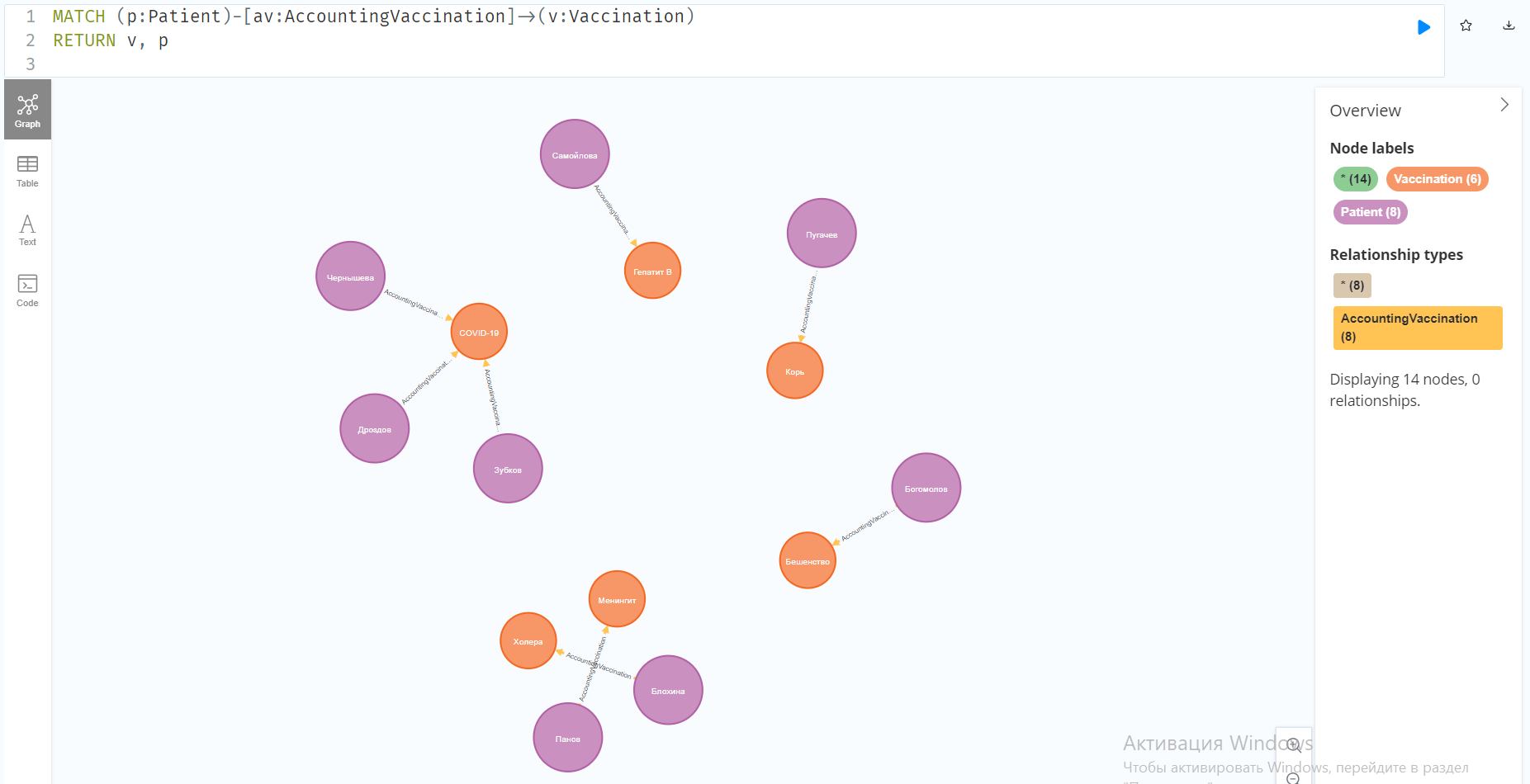


Рисунок 7 – Список вакцин, которые пациенты сделали в поликлинике

1. Вывести всех пациентов, у которых выработался иммунитет после вакцинации COVID-19.

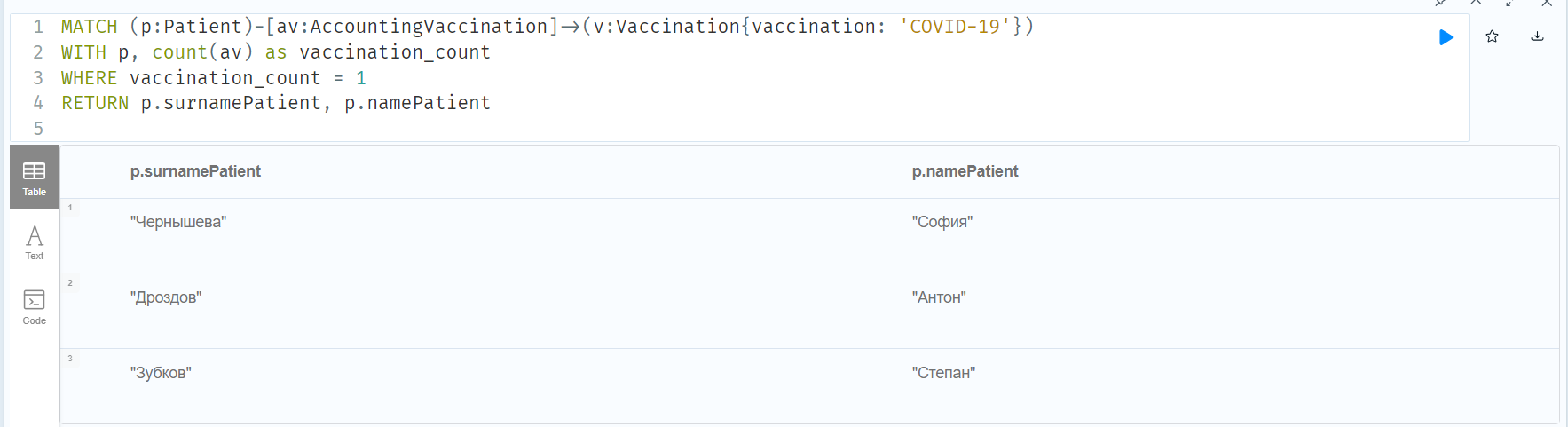
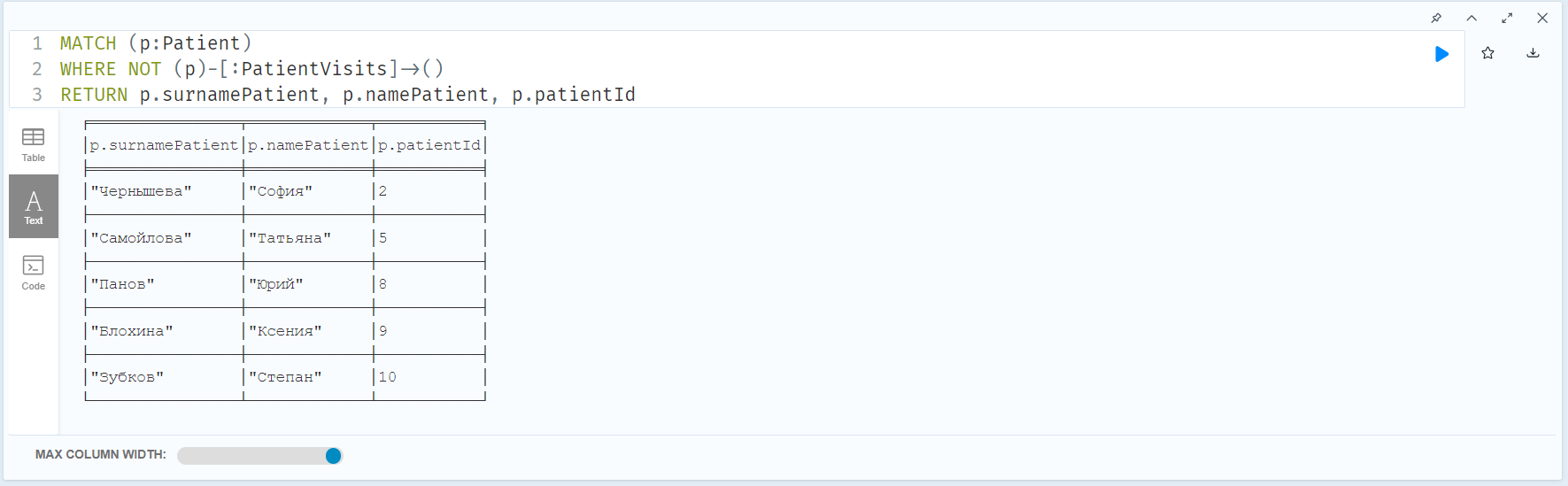


Рисунок 8 – список пациентов, у которых выработался иммунитет

1. Вывести всех пациентов, которые не болели и не приходили в поликлинику.

Рисунок 9 – Список пациентов, которые ни разу не болели

1. Вывести список болезней, которыми болели больше одного раза.

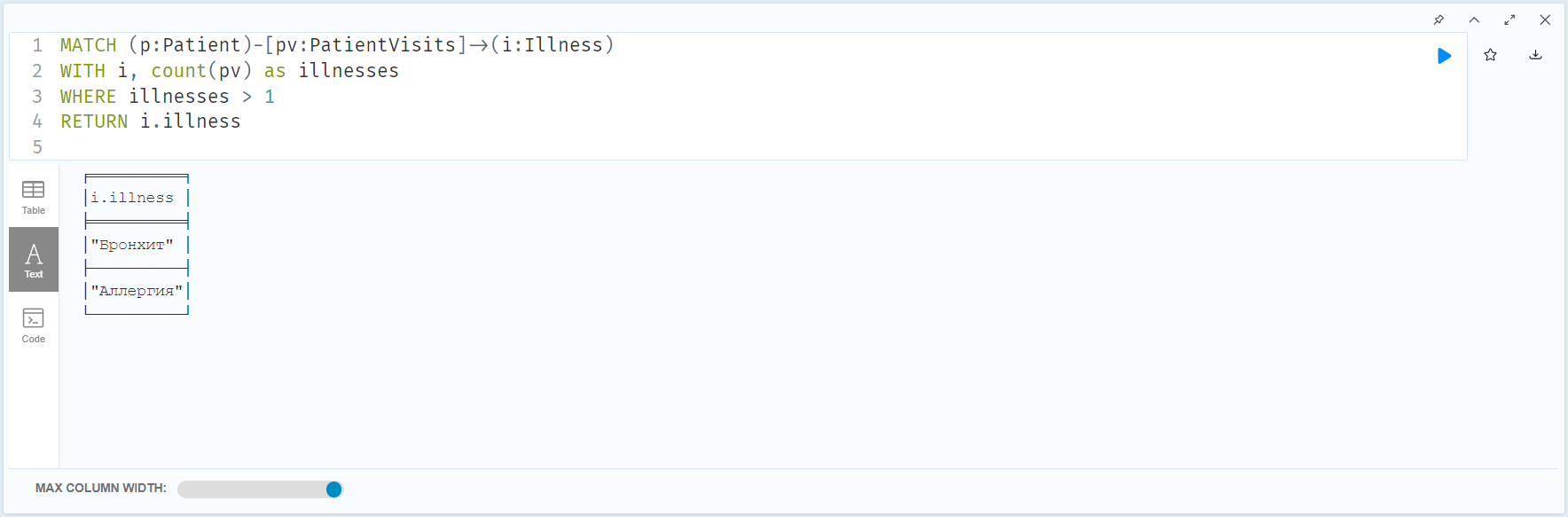


Рисунок 10 – Список болезней, которыми болели несколько раз

1. Посчитать общее количество случаев заболеваний различными болезнями за все время в поликлинике.

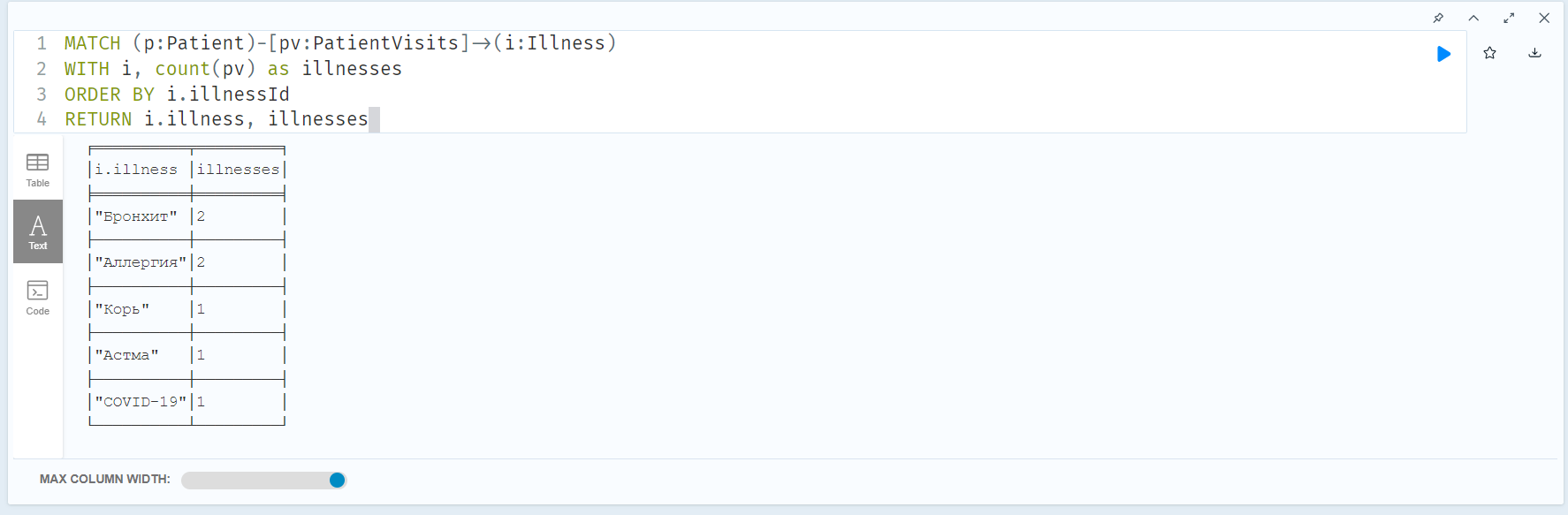


Рисунок 11 – Список общего количества случаев заболеваний различными болезнями

1. Вывести список пациентов, не достигших совершеннолетия.

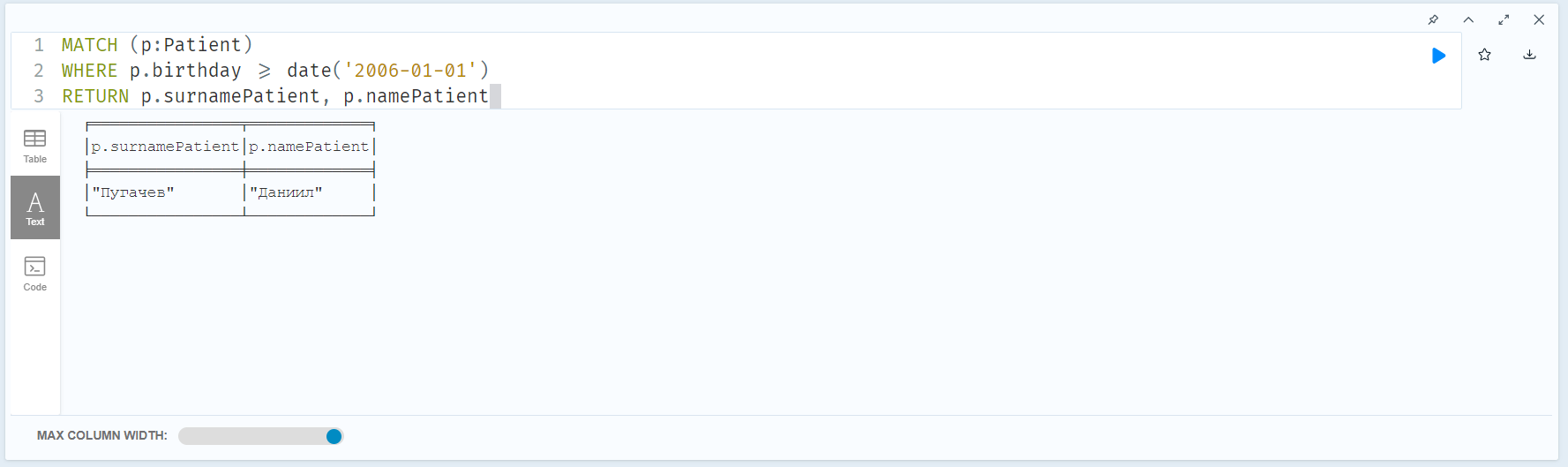


Рисунок 12 – Список пациентов, не достигших 18-ти лет

1. Получить список пациентов, которые болели в октябре.

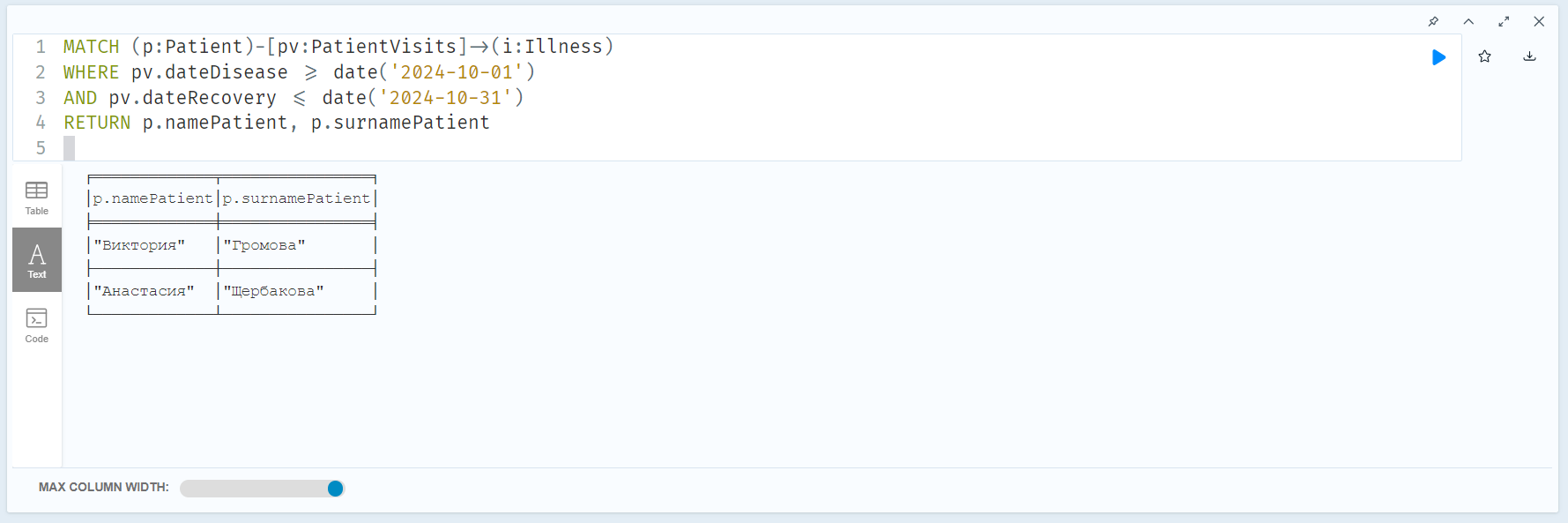
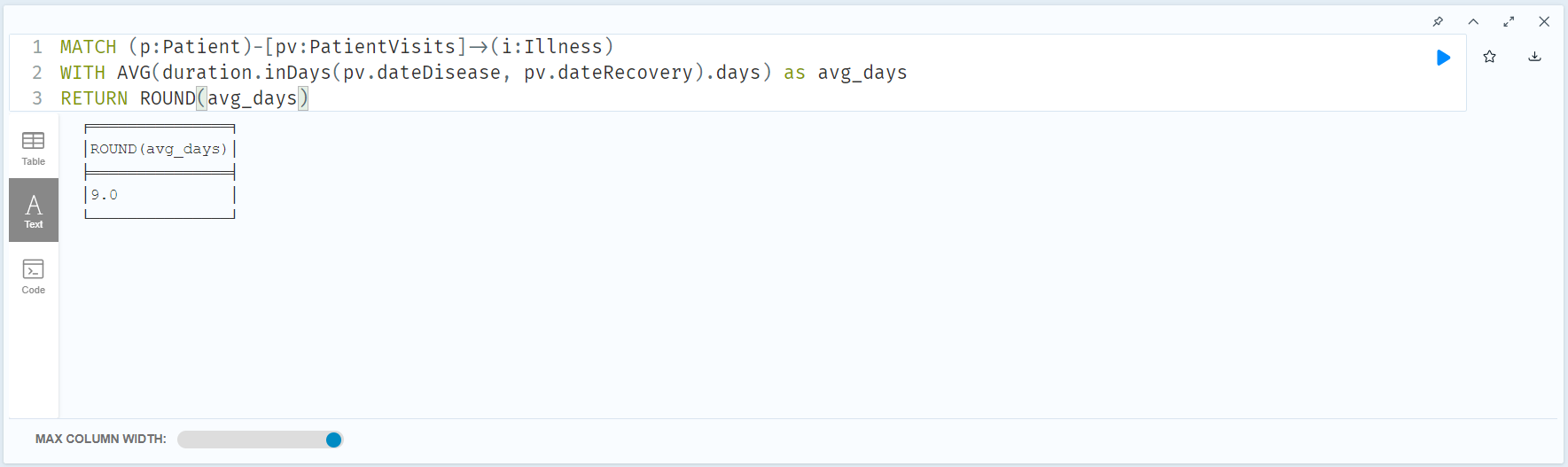


Рисунок 13 – Список пациентов, болевших в октябре

1. Найти среднее количество дней, которые болеют пациенты.

Рисунок 14 – Среднее количество дней, которые болеют пациенты.

**Вывод**

Таким образом, выполняя данную лабораторную работу, получили практические навыки работы с графической базой данных Ne04j. С помощью Chyper-кода научились создавать графическую БД и заполнять ее данными. Реализовали запросы к графической базе данных и зафиксировали результаты их выполнения.