МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 3

з дисципліни “Бази даних”

Виконав
студент III курсу
групи КП-81
Ладуда Данило Володимирович
(*прізвище, ім’я, по батькові*)

Зарахована
“____” “_____” 20___ р.
викладачем
Петрашенко Андрієм Васильовичем (*прізвище, ім’я, по батькові*)

варіант № 9

Київ 2021

Мета роботи: здобуття практичних навичок створення програм, орієнтованих на використання графової бази даних Neo4J за допомогою мови Python.

Завдання роботи полягає у наступному: реалізувати можливості формування графової бази даних в онлайн-режимі на основі модифікованої програми лабораторної роботи №2. На основі побудованої графової бази даних виконати аналіз сформованих даних.

Окремі програмні компоненти та вимоги до них

1. Інфраструктура лабораторної роботи №2:
   1.1. Redis server.
   1.2. Програма емуляції активності користувачі (вхід/вихід, відправка/отримання повідомлення).
   1.3. Виконувач задач (Worker).
2. Сервер Neo4J.
3. Інтерфейс користувача Neo4J.

Порядок виконання роботи

1. В ЛР№2 залишити єдиний режим роботи - емуляція активності.
2. Внести доповнення у програму ЛР№2 шляхом додавання у повідомлення тегу або тегів з переліку, заданого у вигляді констант, обраних студентом.
3. Встановити сервер Neo4J Community Edition.
4. Розробити схему бази даних Neo4J для збереження інформації про активності користувачів (вхід/вихід, відправлення/отримання повідомлень) та Worker (перевірка на спам). Визначити вузли та зв’язки між ними на графі.
5. Розширити функціональність ЛР№2 шляхом збереження будь-якої активності (див. п. 4) у базу даних Neo4J у момент збереження даних у Redis.
6. У програмі “Інтерфейс користувача Neo4J” виконати і вивести результат наступних запитів до сервера Neo4J:

   6.1. Задано список тегів (*tags*). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів *tags*.

   6.2. Задано довжину зв’язку N - кількість спільних повідомлень між користувачами. Знайти усі пари користувачів, що мають зв’язок довжиною N через відправлені або отримані повідомлення. Наприклад,

якщо користувач A відправив повідомлення користувачу B, а B відправив повідомлення C, то довжина зв’язку між A і C є N=2.

6.3. Задано два користувача. Знайти на графі найкоротший шлях між ними через відправлені або отримані повідомлення.

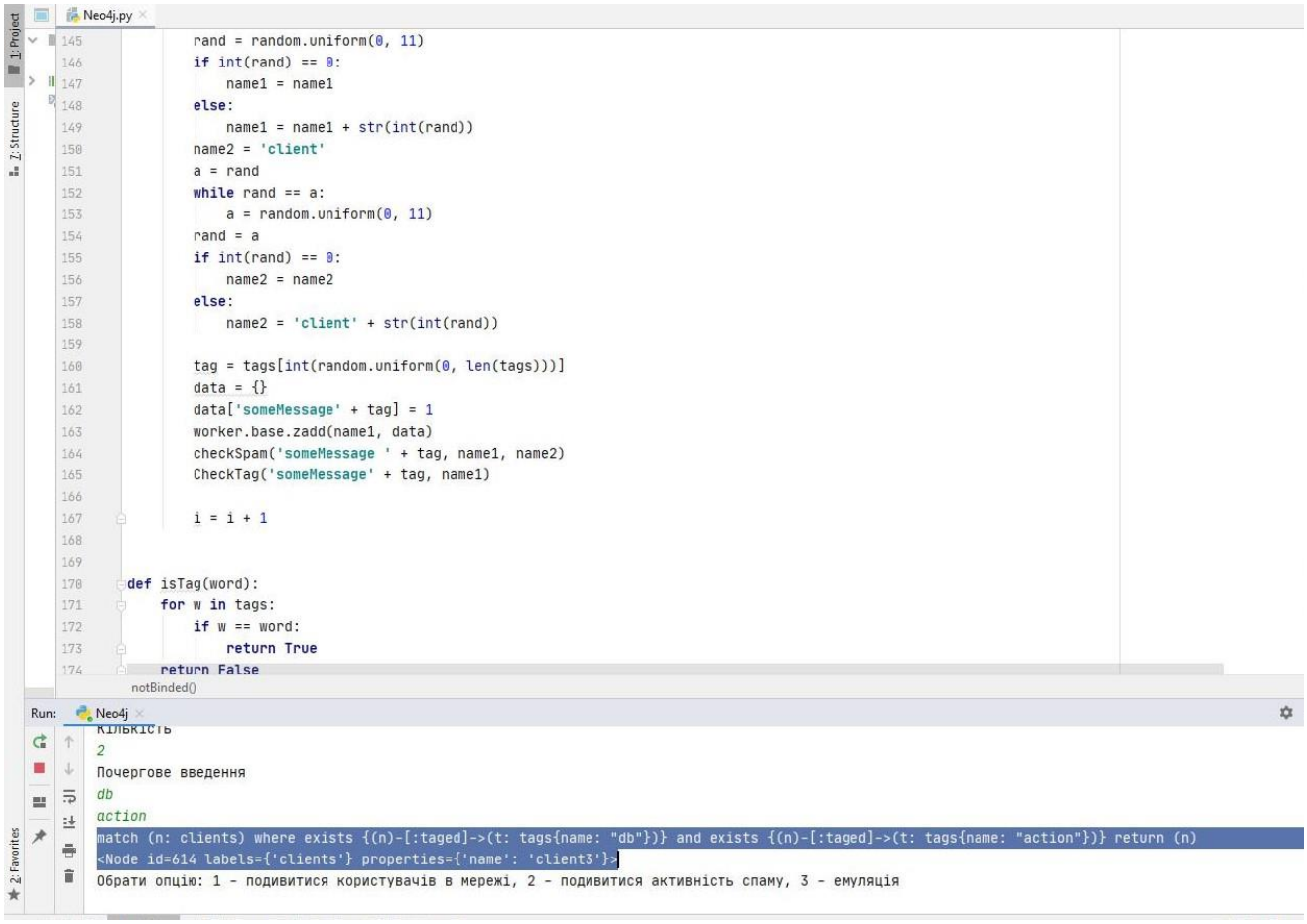
6.4. Знайти авторів повідомлень, які пов’язані між собою лише повідомленнями, позначеними як “спам”.

6.5. Задано список тегів (*tags*). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів tags, але ці користувачі не пов’язані між собою.

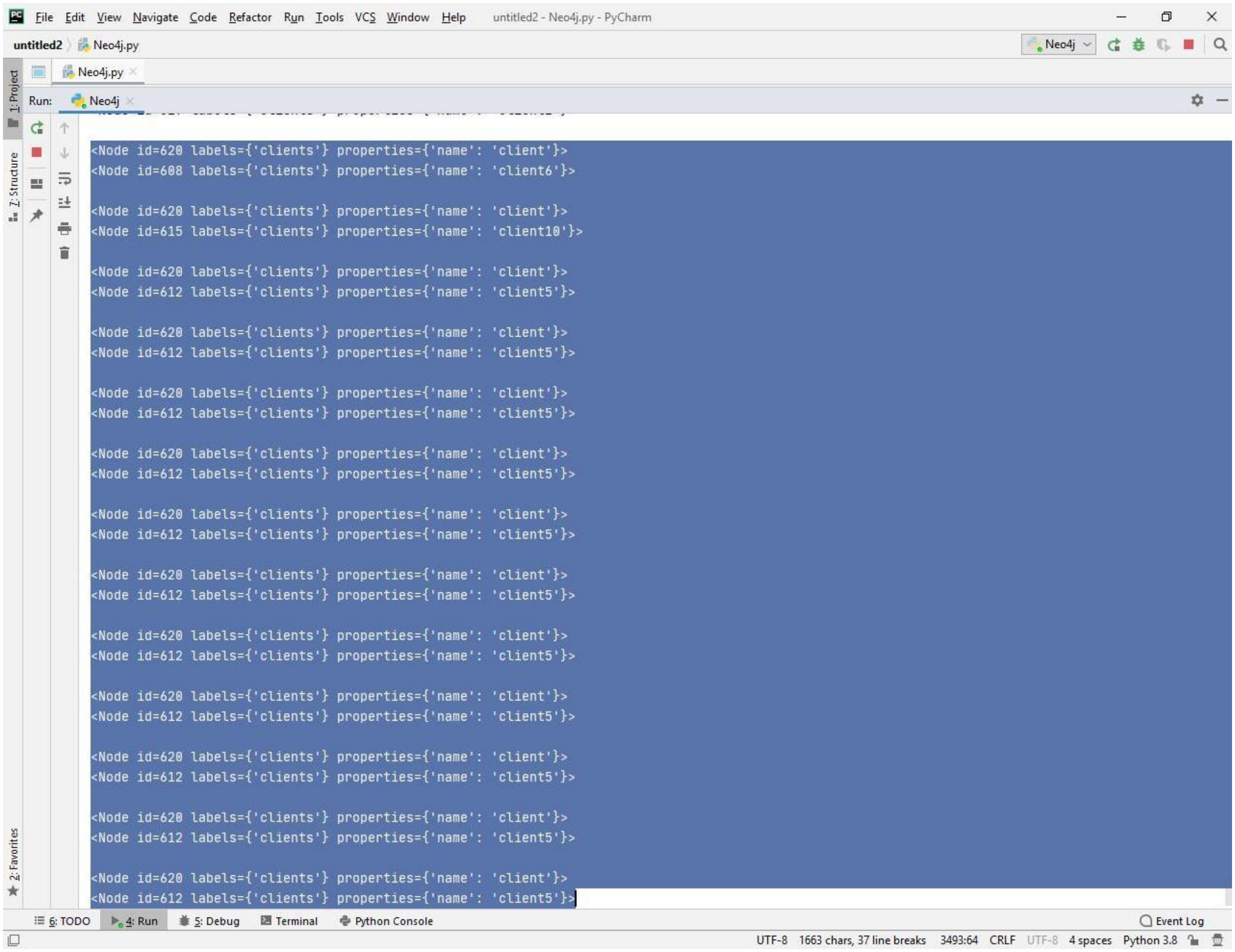
Посилання на репозиторій: https://github.com/ladudanil/db_2term

Результати виконання завдання:

6.1. Задано список тегів (tags). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів tags. tags = [“db”, “action”] match (n: clients) where exists {(n)-[:taged]->(t: tags{name: "action"})} and exists {(n)-[:taged]->(t: tags{name: "db"})} return (n)

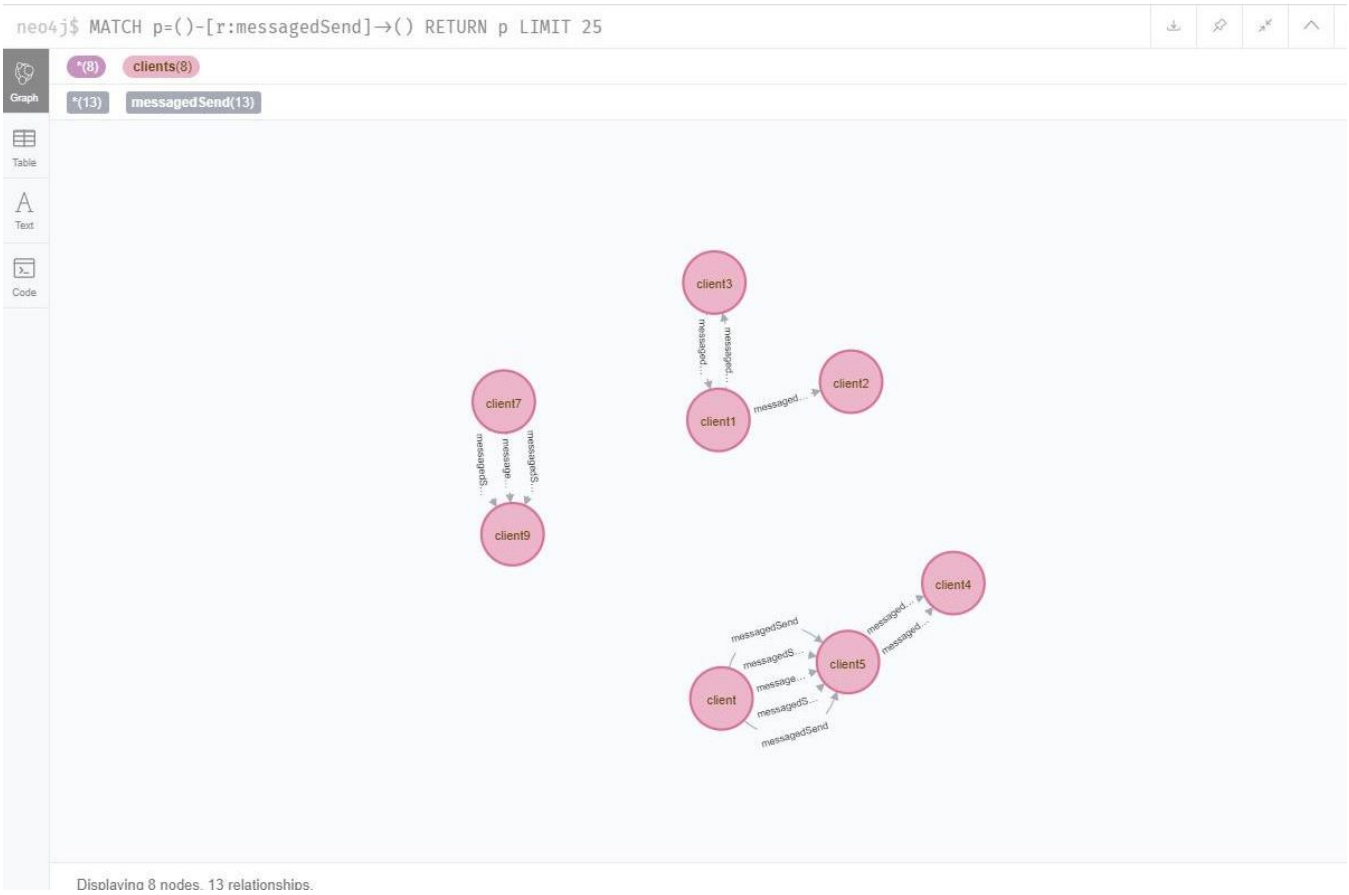


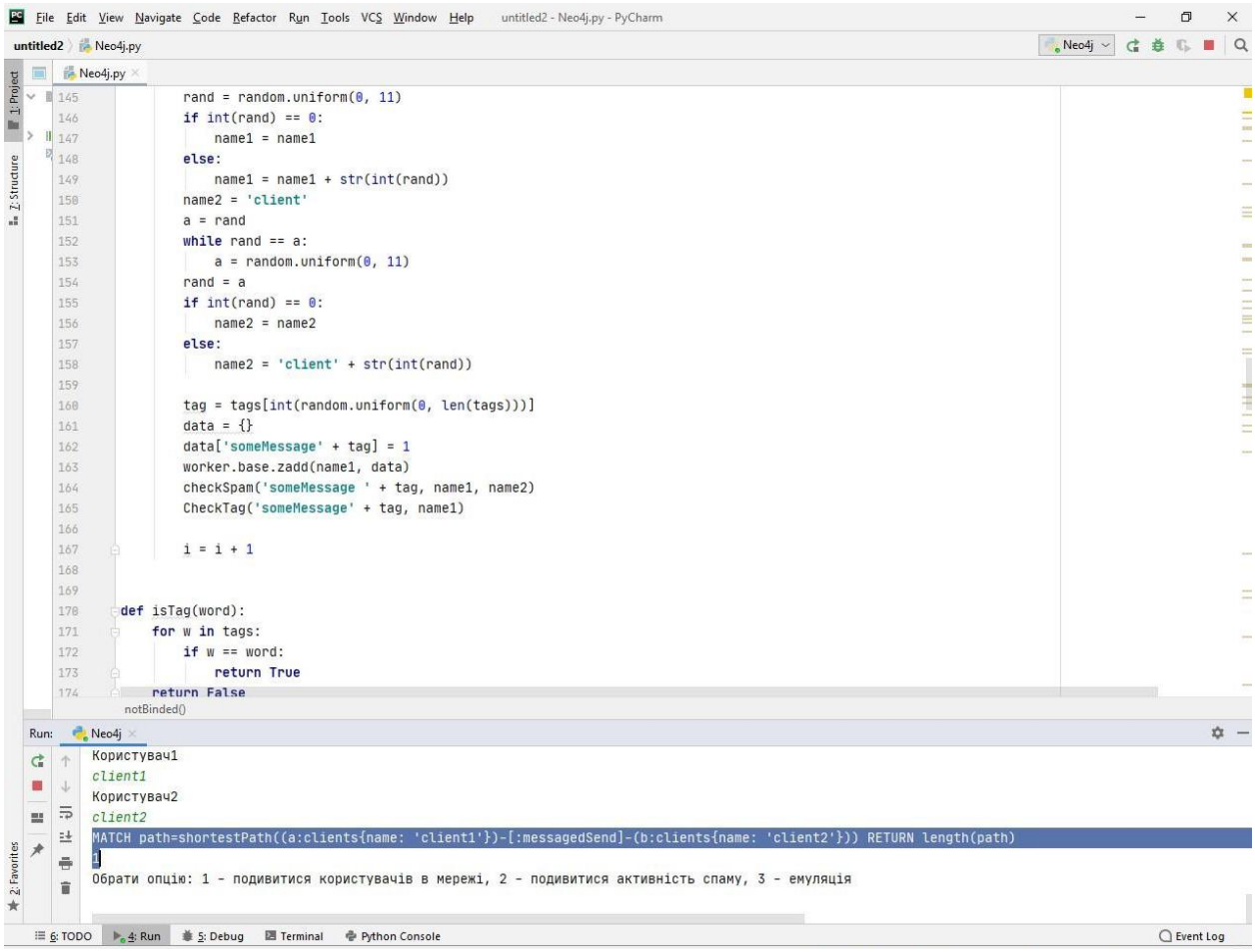
6.2. Задано довжину зв’язку N - кількість спільних повідомлень між користувачами. Знайти усі пари користувачів, що мають зв’язок довжиною N через відправлені або отримані повідомлення. n = 3 MATCH (a:clients)-[messagedSend*3]-(b:clients) RETURN (a), (b)



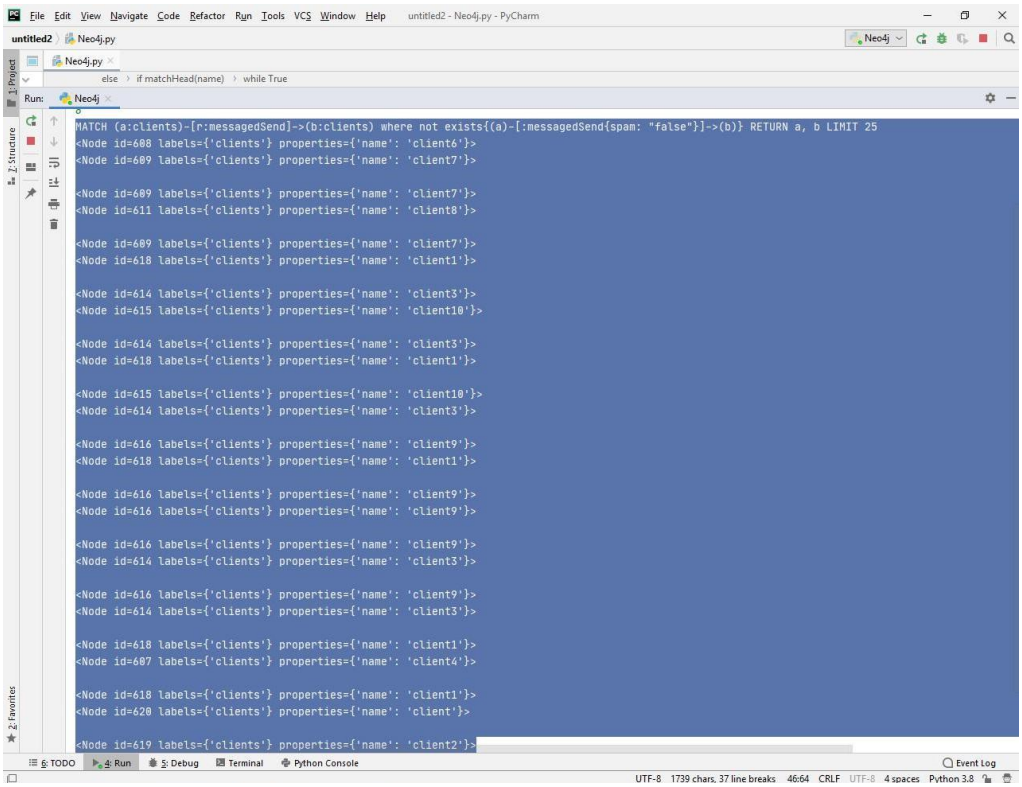
6.3. Задано два користувача. Знайти на графі найкоротший шлях між ними через відправлені або отримані повідомлення. Користувачі - client1, client2

MATCH path=shortestPath((a:clients{name: 'client1'})-[:messagedSend]-(b:clients{name: 'client2'})) RETURN length(path)



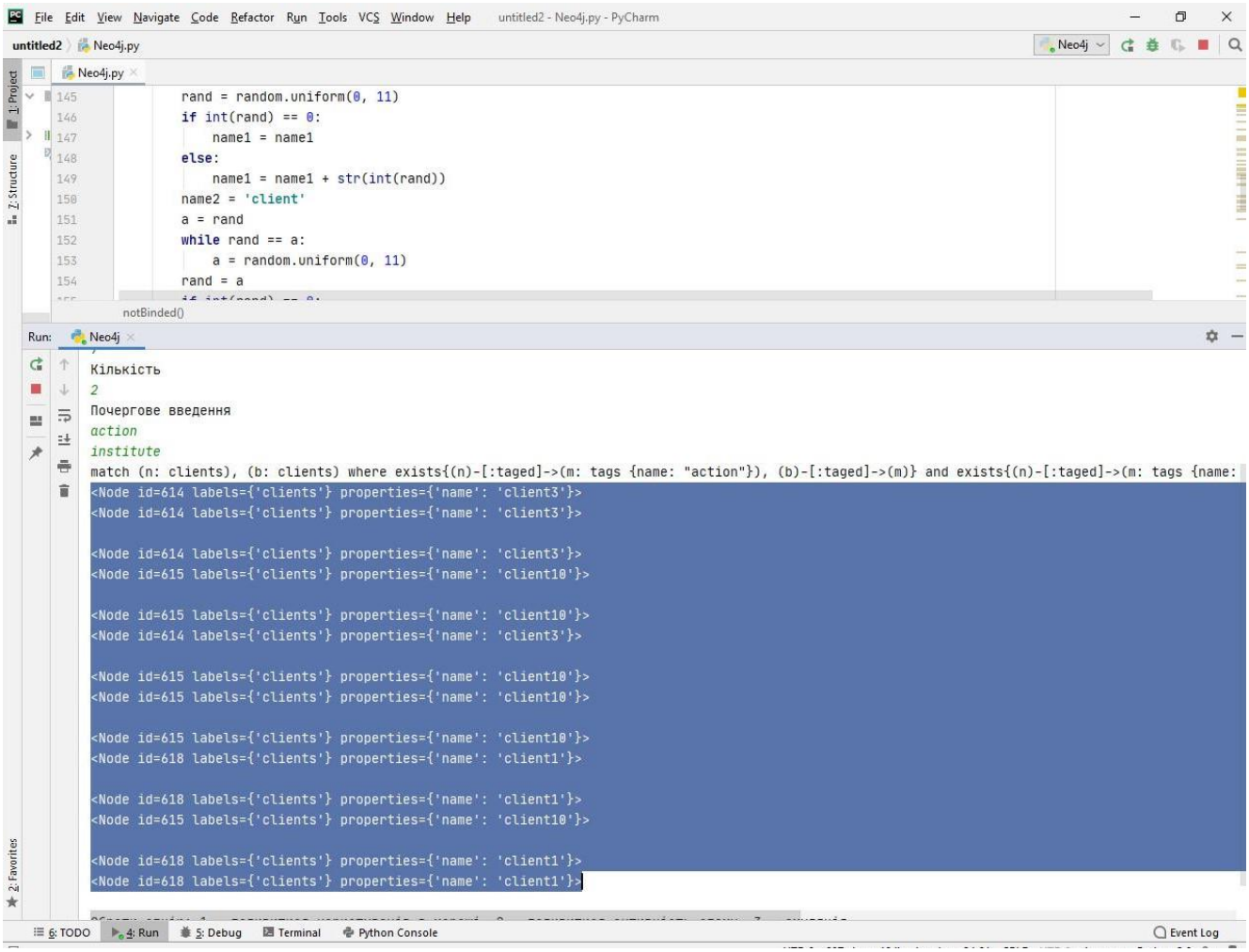


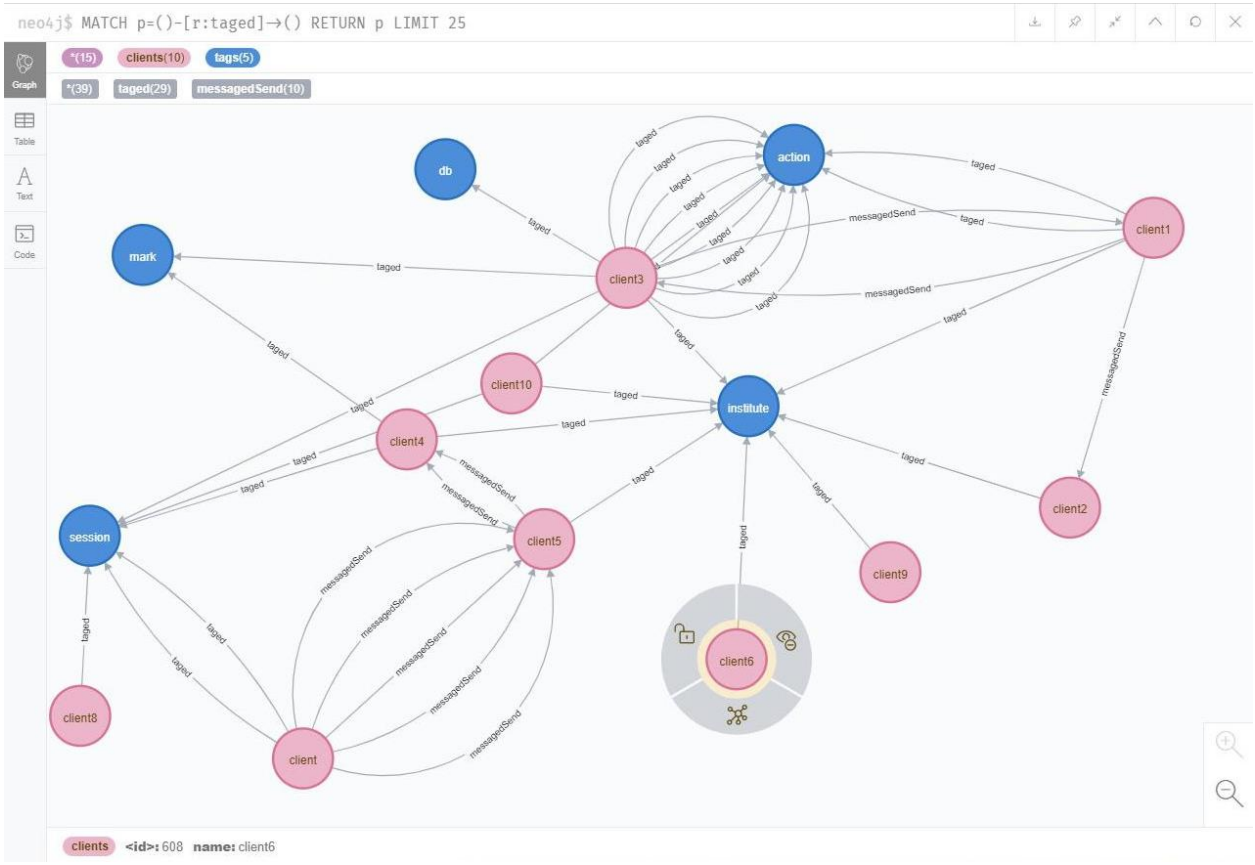
6.4. Знайти авторів повідомлень, які пов’язані між собою лише повідомленнями, позначеними як “спам”. MATCH (a:clients)-[r:messagedSend]->(b:clients) where not exists{(a)-[:messagedSend{spam: "false"}]->(b)} RETURN a, b



6.5. Задано список тегів (tags). Знайти всіх користувачів, що відправили або отримали повідомлення з набором тегів tags, але ці користувачі не пов’язані

між собою. tags = [“action”, “institute”] match (n: clients), (b: clients) where exists{(n)-[:taged]->(m: tags {name: "action"}), (b)-[:taged]->(m)} and exists{(n)-[:taged]->(m: tags {name: "institute"}), (b)-[:taged]->(m)} and not exists {(n)-[:messagedSend]-(b)} return (n), (b)




neo4j$ MATCH p=()-[r:taged]→() RETURN p LIMIT 25
Graph
Table
Text
Code
*(15)
clients(10)
tags(5)
*(39)
taged(29)
messagedSend(10)
db
action
client1
mark
client3
client10
institute
client4
client2
session
client5
client9
client6
client8
client
taged
messagedSend
clients <id>: 608 name: client6