1. Introduction

You will be applying the knowledge that you've gained regarding PostgreSQL and Neo4j to a real-world style problem, tracking information for a hospital!

I will be giving you a high-level description of required capabilities, but I encourage you to **expand on these requirements** and add functionality and capabilities to your database design beyond what I'll be describing below.

2. Objectives

By the end of this project, you will be able to:

1. Design a database for housing real-world style data in PostgreSQL
2. Design a database for housing real-world style data in Neo4j
3. Write scripts to load data into your databases

3. What to Do

The purpose of this project is to put to work the knowledge you've gained regarding PostgreSQL and Neo4j by implementing a database design in both databases, solving the same problem twice! You will then load data into your databases and run queries against them to compare how well both databases solved the problem, and then write up a summary detailing which database you considered the easiest to use and most applicable to the problem space.

In this project you will be implementing a database for Small-town Hospital, a local hospital in town. Small-town Hospital has a lot of information to store about their doctors, patients, illnesses, and treatments, and is hoping you can help them out!!

* I will not be specifying specific table structures for this project. Instead I will give general guidelines that you will need to use to develop the correct tables in Postgres, and Labels/Relationships in Neo4j:
  + A **Doctor**has a **name** and sees **many Patients**
  + A **Patient** has a **name**and can have **many Illnesses.** A Doctor can be a Patient
  + An **Illness** has a **name**
  + A **Patient**can receive **many Treatments**
  + A **Treatment**has a name
* **NOTE:**Be sure to include the SQL needed to create your tables for PostgreSQL in your submission. For Neo4j, describe the labels and relationships that you intend to use to represent the data.
* Now that we have the schema in place, we need to insert a bunch of data and run a couple of queries!
  + Create load scripts for each database that will insert **at least 10000 unique patients, 100 unique doctors, 1000 unique illnesses, and 750 unique treatments. Every patient should be seeing 1-5 doctors and should have between 0-3 different illnesses. Any patients that have an illness should be receiving at least one treatment. Doctors have a 35% chance of being a patient themselves, in which case the patient rules apply to them.**
  + Once you have the data inserted, design queries for Postgres and Neo4j that query for a patient's doctors, illnesses, and treatments. Is it possible to return all of this information in one query, or are multiple requires required?
* After you've completed all the tasks above, create a summary writeup comparing the pros and cons of using PostgreSQL and Neo4j to solve this specific problem set. What went well? What didn't? Are there features unique to one database or another that helped you? Describe all of this in your write up. This write up is one of the most important points of the project! Making sure you understand what database is right for the job is one of the most important outcomes of taking this course, so please take some time when writing up your thoughts.

Answer the following five questions about the data. If you complete only the Cypher queries and not the Postgres queries, I will give you 1 point per question (5 points total):

1. How many doctors are treating doctors?
2. What's the count of how many patients have each kind of illness?
3. What's the doctor with the most patients?
4. Which doctor is treating the largest number of unique illnesses?
5. What illness is being treated with the largest number of unique treatments?