

Introduction

There is a plethora of individuals who seek a personalized blend of fitness programs and nutritional supplements put together just for them. The goal of using user data visualized as a graph is to recommend fitness programs, nutritional supplements and other users that would be personalized to a user's persona.

This recommendation system does exactly that, it demonstrates the personalized recommendations and connected data benefits of transforming the existing data into a Neo4j graph database.

Introduction to the Dataset

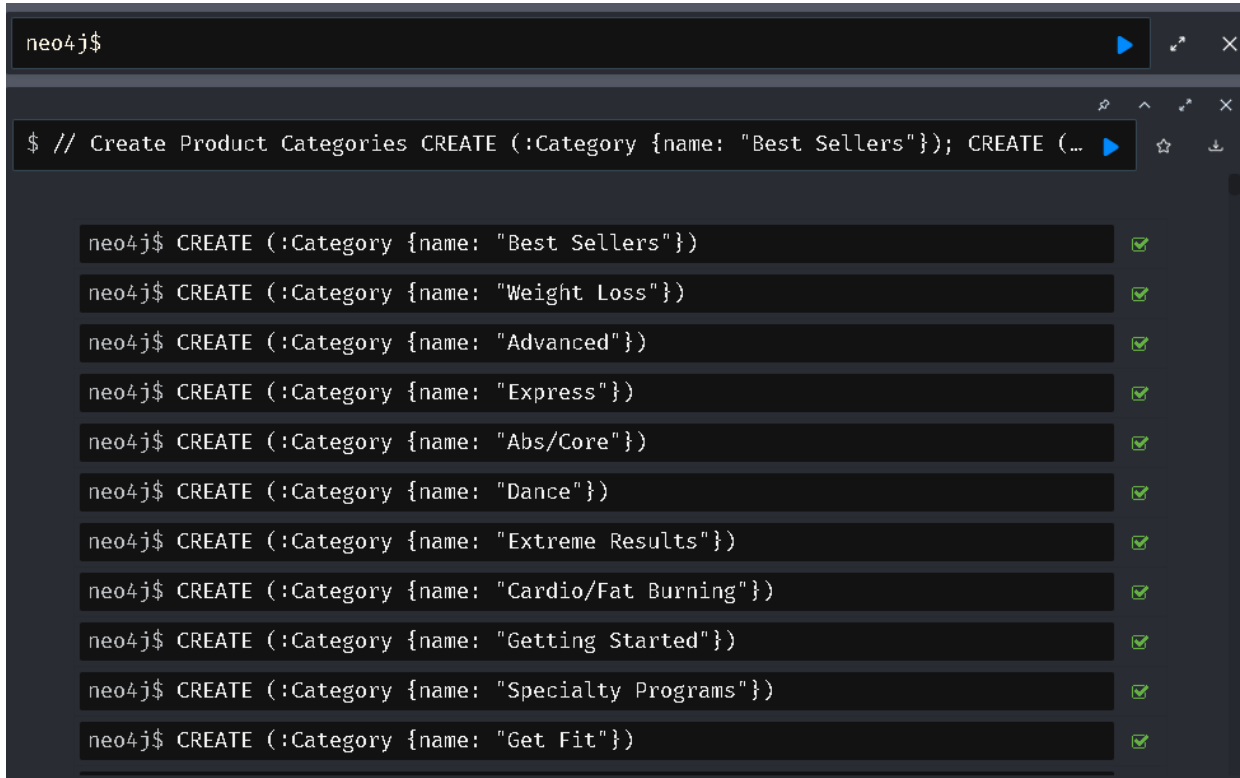
The dataset used contains data pertaining to the user, such as:

- Workout Goals
- Eating Goals
- Muscle Groups
- Workout Types
- Supplement Types

All attributes are closely linked together.

Neo4j Commands and Analysis

1. Importing* the dataset into Neo4j.



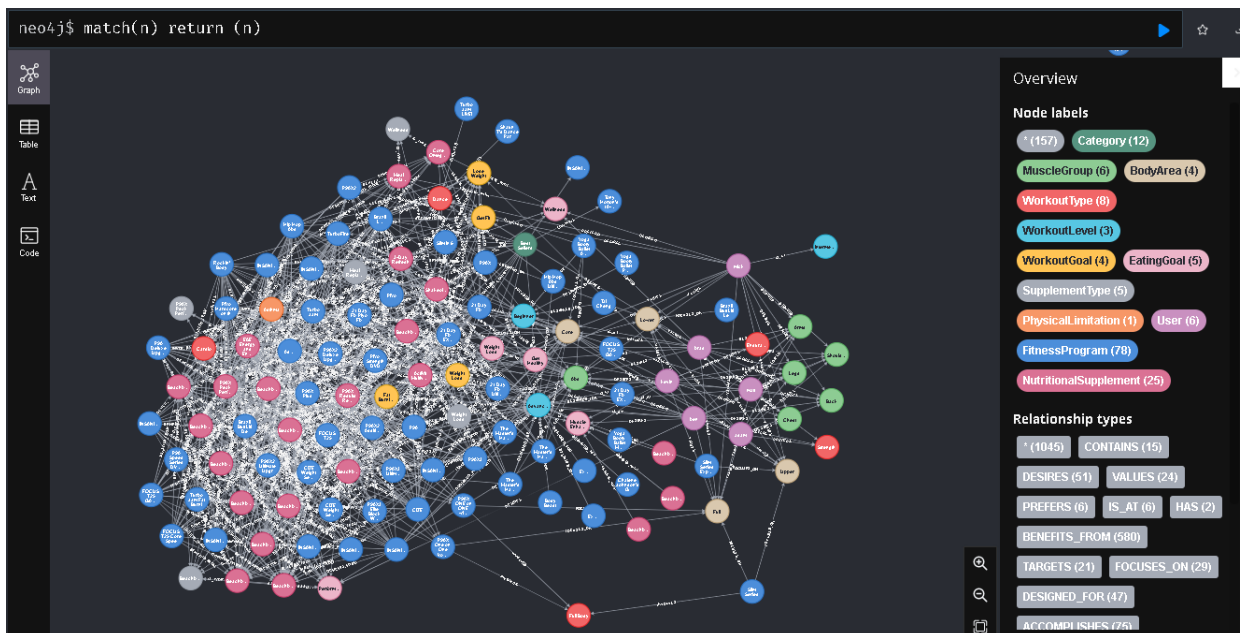
```
neo4j$  
$ // Create Product Categories CREATE (:Category {name: "Best Sellers"}); CREATE (...  
neo4j$ CREATE (:Category {name: "Best Sellers"})  
neo4j$ CREATE (:Category {name: "Weight Loss"})  
neo4j$ CREATE (:Category {name: "Advanced"})  
neo4j$ CREATE (:Category {name: "Express"})  
neo4j$ CREATE (:Category {name: "Abs/Core"})  
neo4j$ CREATE (:Category {name: "Dance"})  
neo4j$ CREATE (:Category {name: "Extreme Results"})  
neo4j$ CREATE (:Category {name: "Cardio/Fat Burning"})  
neo4j$ CREATE (:Category {name: "Getting Started"})  
neo4j$ CREATE (:Category {name: "Specialty Programs"})  
neo4j$ CREATE (:Category {name: "Get Fit"})
```

*Dataset used was in the form of create commands.

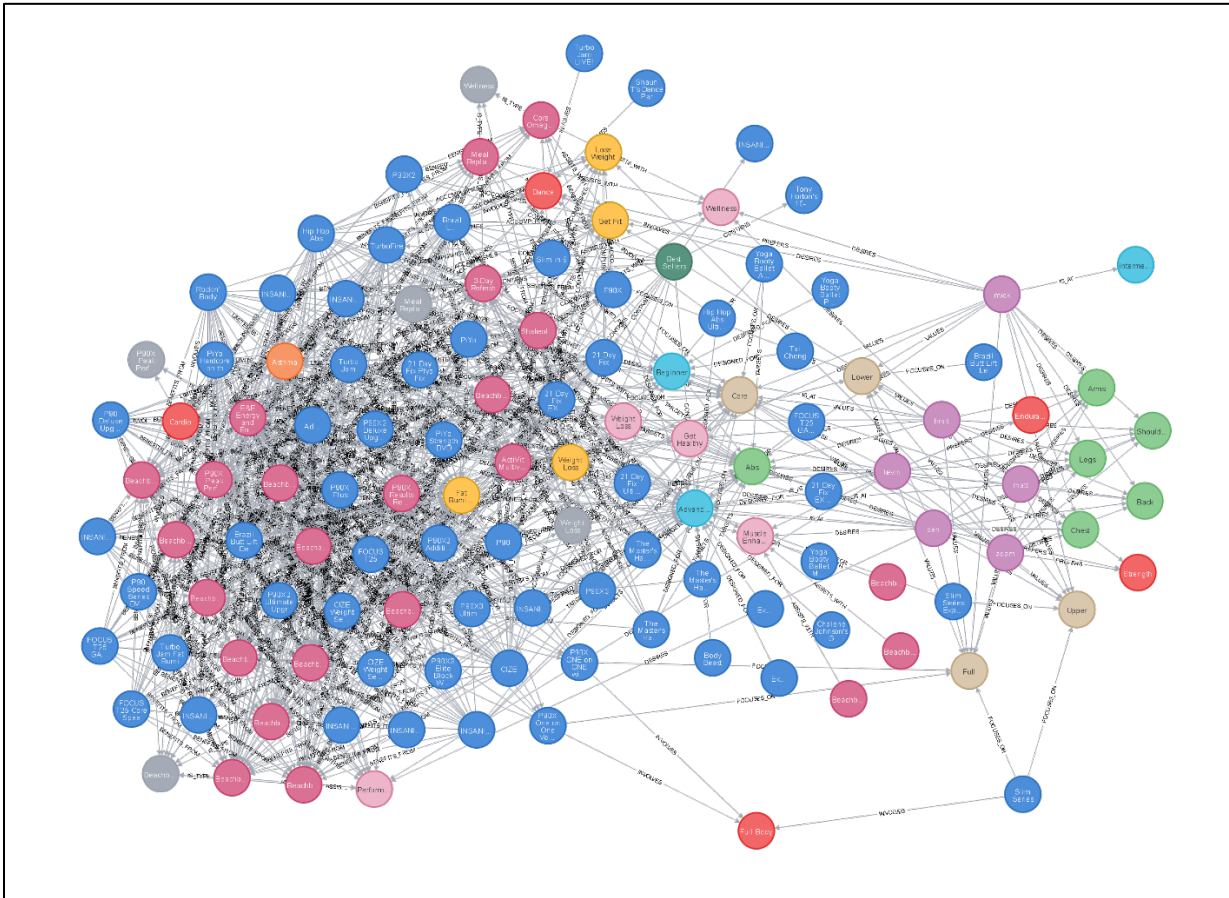
Graph Theory and Its Applications: Assignment-4

```
neo4j$  
$ // Create Constraints CREATE CONSTRAINT FOR (c:Category) REQUIRE c.name IS UNIQUE...  
neo4j$ CREATE CONSTRAINT FOR (c:Category) REQUIRE c.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (mg:MuscleGroup) REQUIRE mg.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (ba:BodyArea) REQUIRE ba.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (wt:WorkoutType) REQUIRE wt.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (wg:WorkoutGoal) REQUIRE wg.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (wl:WorkoutLevel) REQUIRE wl.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (fp:FitnessProgram) REQUIRE fp.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (ns:NutritionalSupplement) REQUIRE ns.name IS UNIQU...  
neo4j$ CREATE CONSTRAINT FOR (ns:EatingGoal) REQUIRE ns.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (ns:SupplementType) REQUIRE ns.name IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (u:User) REQUIRE u.username IS UNIQUE  
neo4j$ CREATE CONSTRAINT FOR (pl:PhysicalLimitation) REQUIRE pl.name IS UNIQUE
```

2. The graph generated out of the dataset.



Graph Theory and Its Applications: Assignment-4



3. a) Distinct nodes in 'FitnessProgram'.

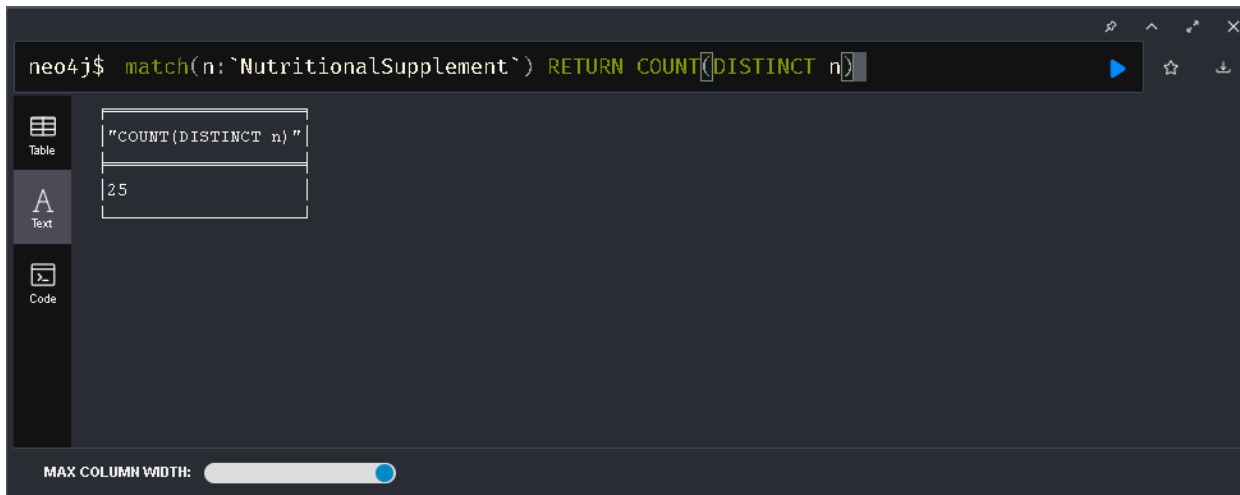
```
neo4j$ match(n:`FitnessProgram`) RETURN COUNT(DISTINCT n)
```

"COUNT(DISTINCT n)"
78

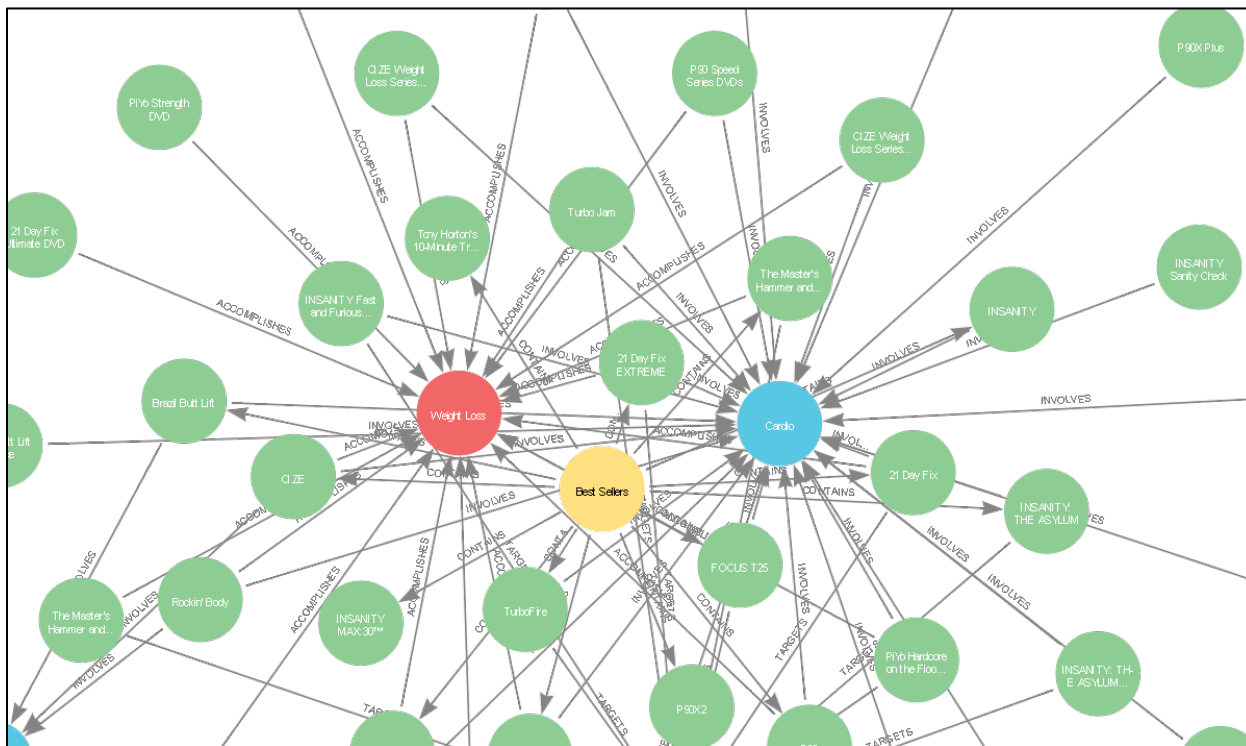
MAX COLUMN WIDTH:

Graph Theory and Its Applications: Assignment-4

3. b) Distinct nodes in 'NutritionalSupplement'.

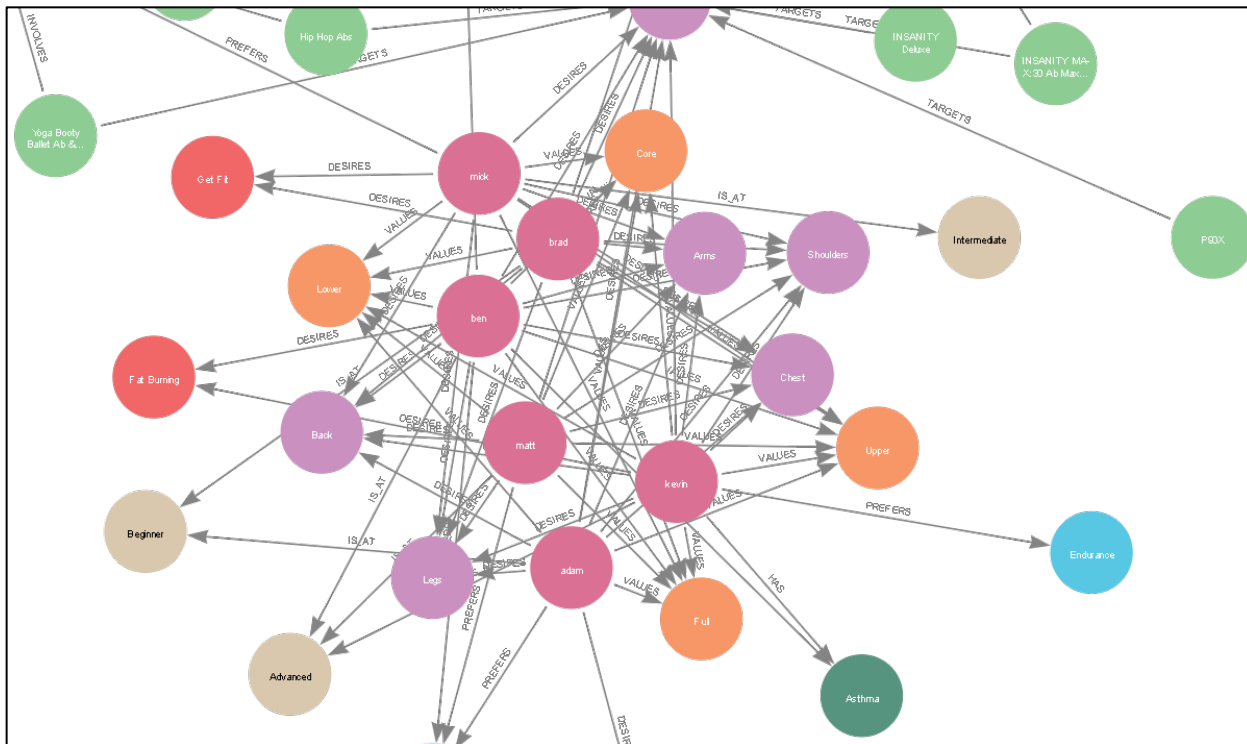


4. a) Relationships: FitnessProgram and Category.



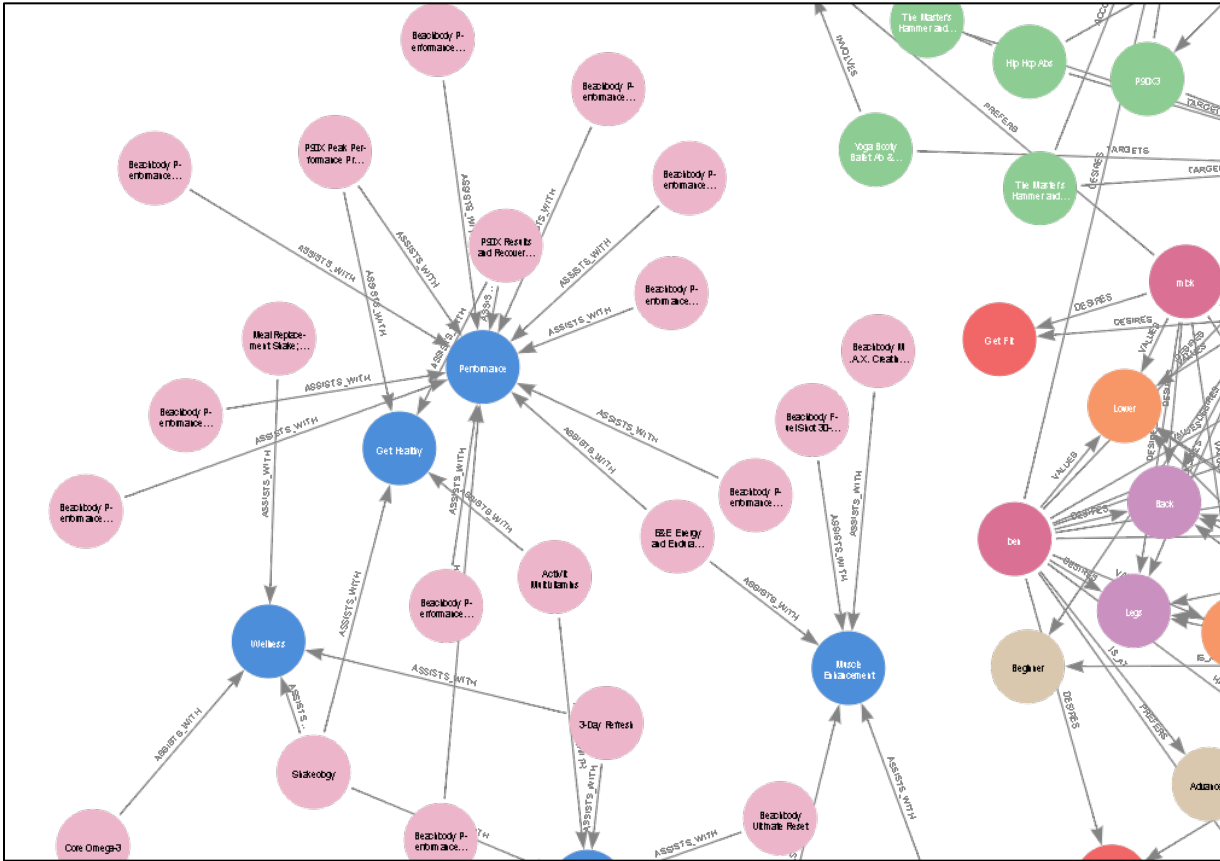
Graph Theory and Its Applications: Assignment-4

4. b) Relationships: MuscleGroup and User.



Graph Theory and Its Applications: Assignment-4

4. c) Relationships: EatingGoals and Supplemets.



Graph Theory and Its Applications: Assignment-4

Initial Analysis

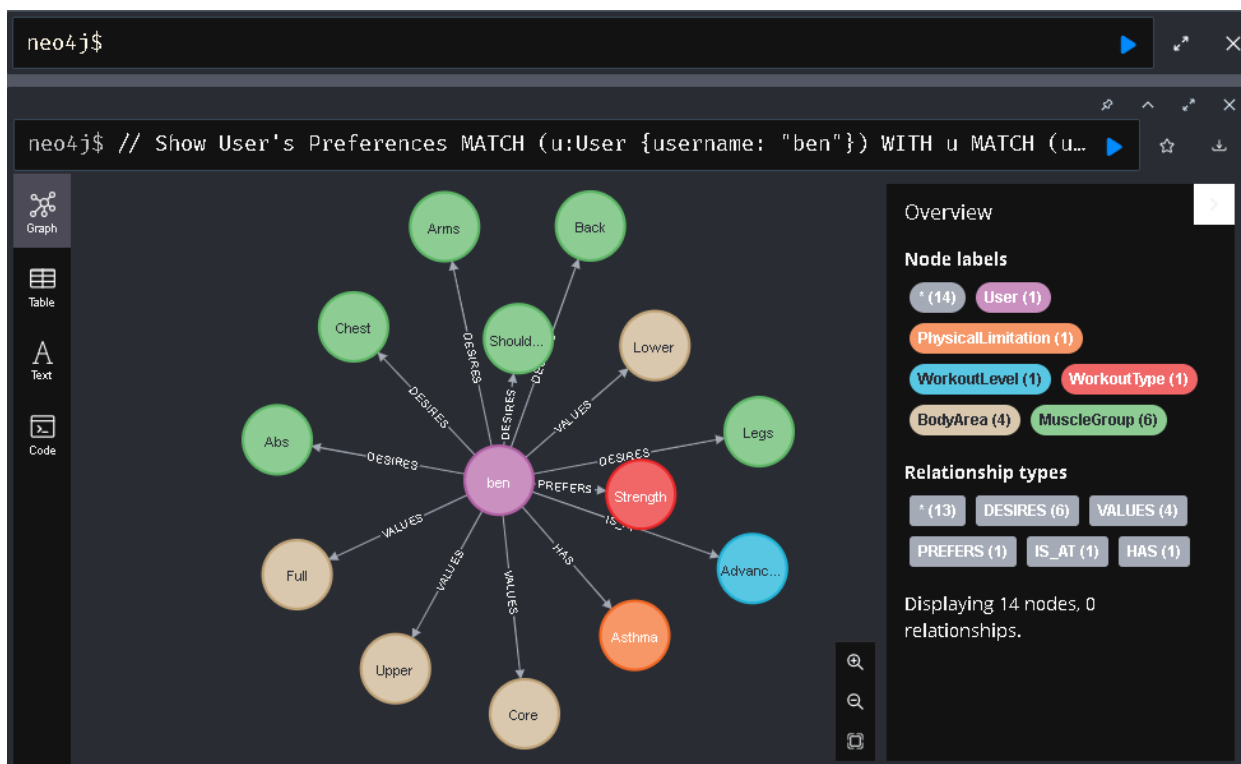
1. Importance the user 'Ben' places on various WorkoutGoals, EatingGoals, MuscleGroups which will be used to recommend workouts and supplements.

```
neo4j$ // Order User's Values by Importance MATCH (u:User {username: "ben"})-[r:VA...
```

	x.name	r.importance
1	"Core"	1.0
2	"Upper"	0.8
3	"Full"	0.7
4	"Lower"	0.4

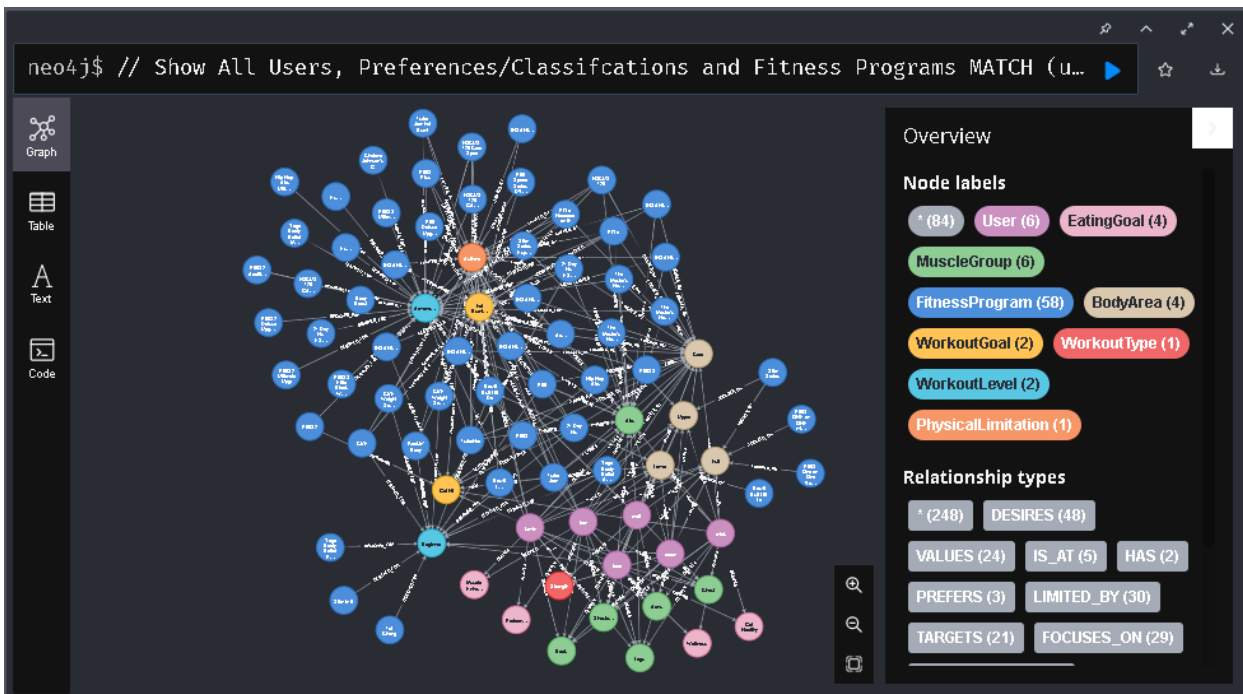
Started streaming 4 records after 11 ms and completed after 13 ms.

2. Moving one level out to the direct information we have available to visualize Ben's preferences.

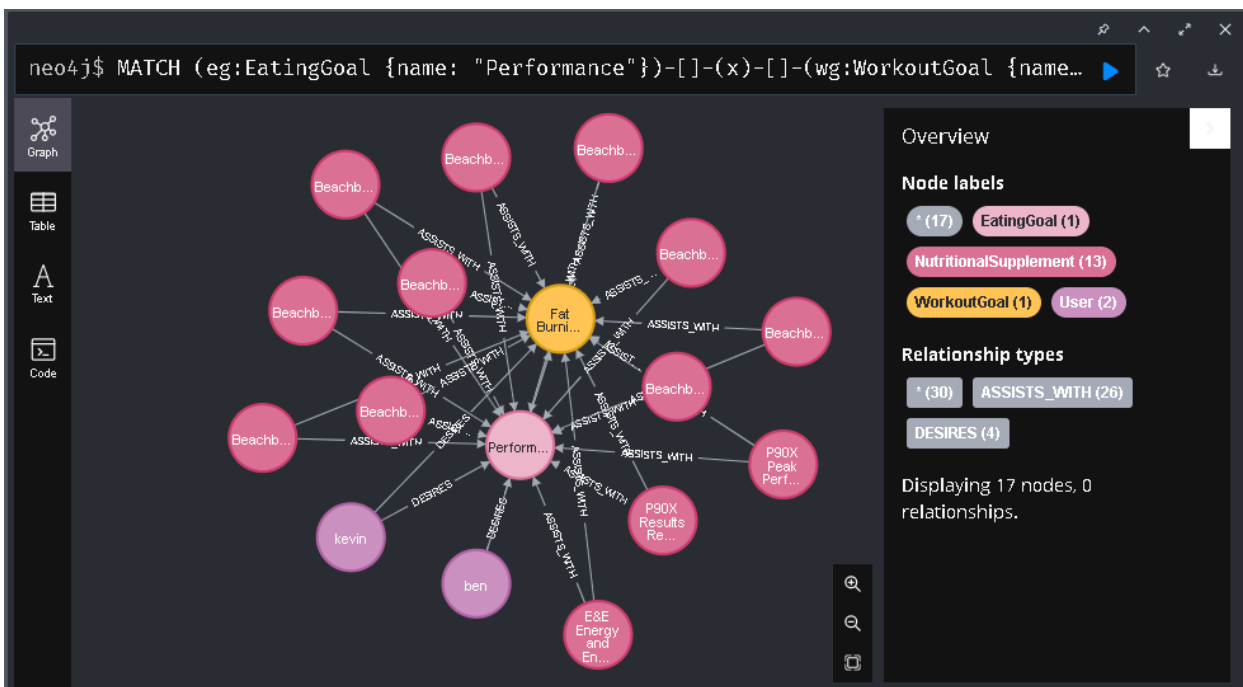


Graph Theory and Its Applications: Assignment-4

3. Similarly we can visualize similar features with all other user's and also see the connected FitnessPrograms.



4. We look at which EatingGoals and WorkoutGoals would be considered parallel, this helps us examine patterns about a user where there is no direct connection.



The Recommendations

A) Content Based Filtering

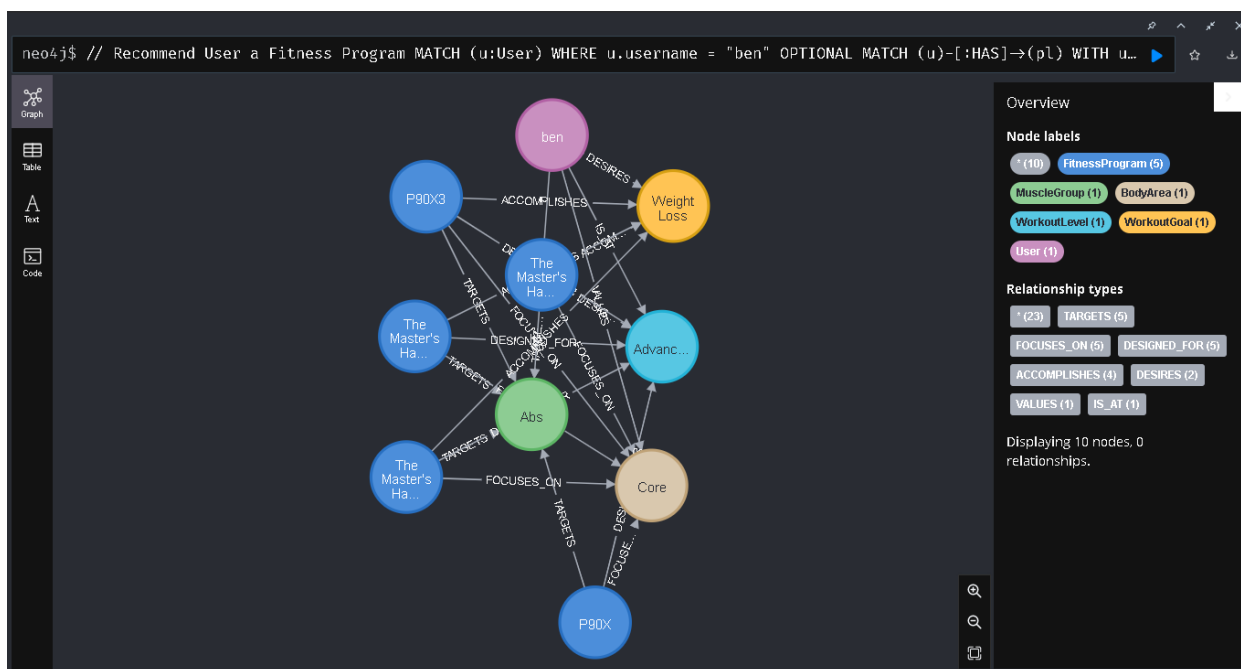
1. Recommend a User a Fitness Program.

We start simple by recommending a User their top five fitness programs.

We then go find all the preferences with the importance weighting. Then we build the traits with their weights and score the significance of the connection between the user and the fitness programs possible to be recommended.

Then we order by that score and limit the return set to 5.

```
// Recommend User a Fitness Program
MATCH (u:User) WHERE u.username = "ben" OPTIONAL MATCH (u)-[:HAS]->(pl)
WITH u, pl MATCH (u)-[r:IS_AT|PREFERS|DESIRES|VALUES]->(x)
WITH u, pl, x, coalesce(r.importance, 0.5) AS importance
MATCH (x)-[]-(x2:FitnessProgram) WHERE NOT (x2)-[:LIMITED_BY]->(pl)
WITH u, x2, collect({name: x.name, weight: importance}) AS traits
WITH u, x2, reduce(s = 0, t IN traits | s + t.weight) AS score
WITH u, x2, score OPTIONAL MATCH (x2)-[]->(x)-[]-(u)
RETURN x2, collect(x) AS x, u, score ORDER BY score DESC LIMIT 5;
```



Graph Theory and Its Applications: Assignment-4

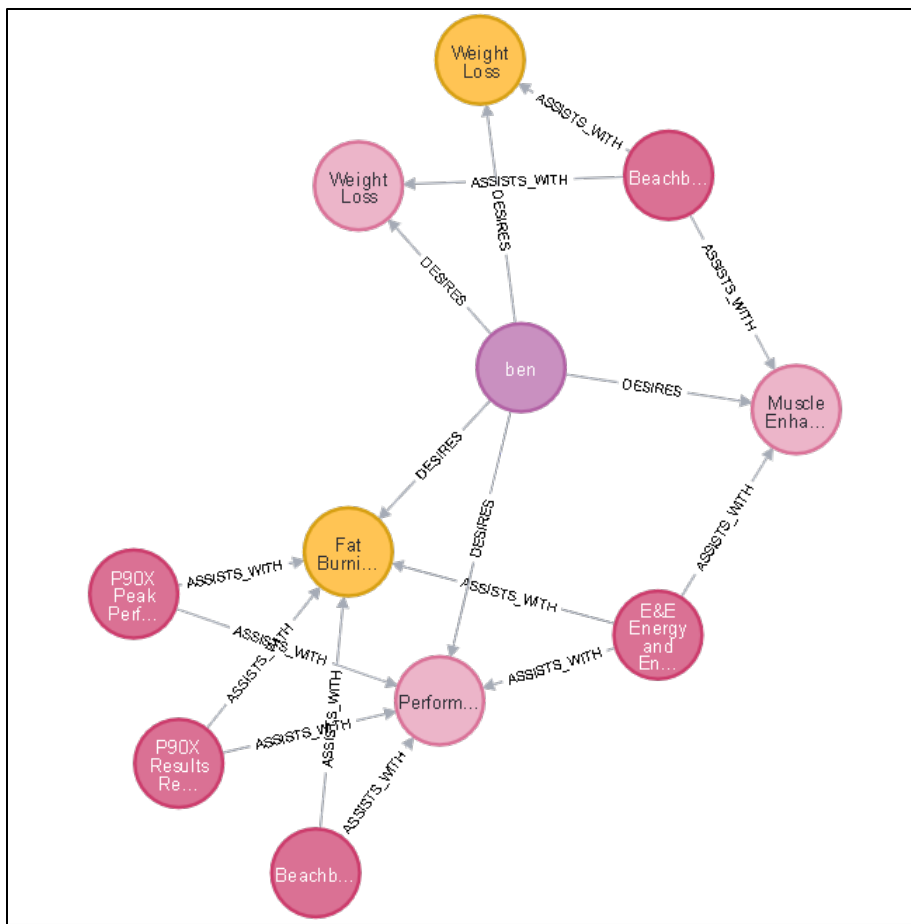
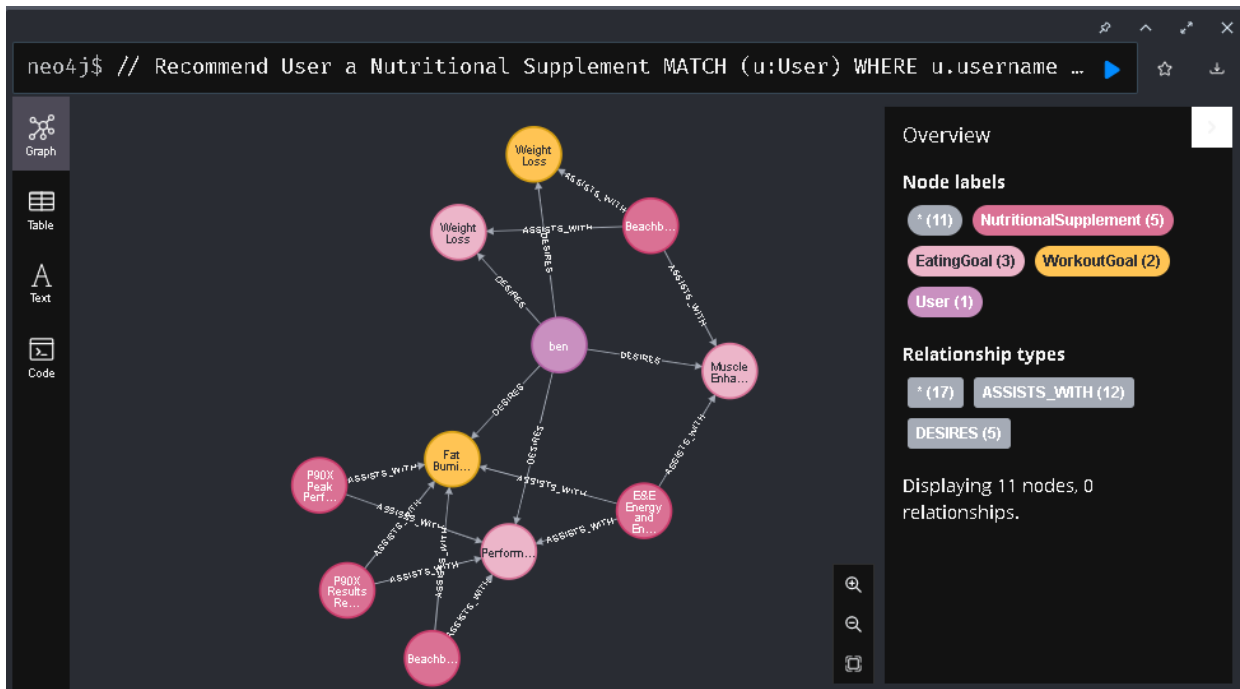


2. Recommend a User a Nutritional Supplement.

The process here is the same as with fitness programs, but is just using nutritional supplements as the thing being recommended. This works because the patterns of relationships are the same across both types.

```
// Recommend User a Nutritional Supplement
MATCH (u:User) WHERE u.username = "ben" OPTIONAL MATCH (u)-[:HAS]->(p1)
WITH u, p1 MATCH (u)-[:IS_AT|PREFERS|DESIRES|VALUES]->(x)
WITH u, p1, x, coalesce(r.importance, 0.5) AS importance
MATCH (x)-[]-(x2:NutritionalSupplement) WHERE NOT (x2)-[:LIMITED_BY]->(p1)
WITH u, x2, collect({name: x.name, weight: importance}) AS traits
WITH u, x2, reduce(s = 0, t IN traits | s + t.weight) AS score
WITH u, x2, score OPTIONAL MATCH (x2)-[]-(x)-[]-(u)
RETURN x2, collect(x) AS x, u, score ORDER BY score DESC LIMIT 5;
```

Graph Theory and Its Applications: Assignment-4



B) Collaborative Filtering

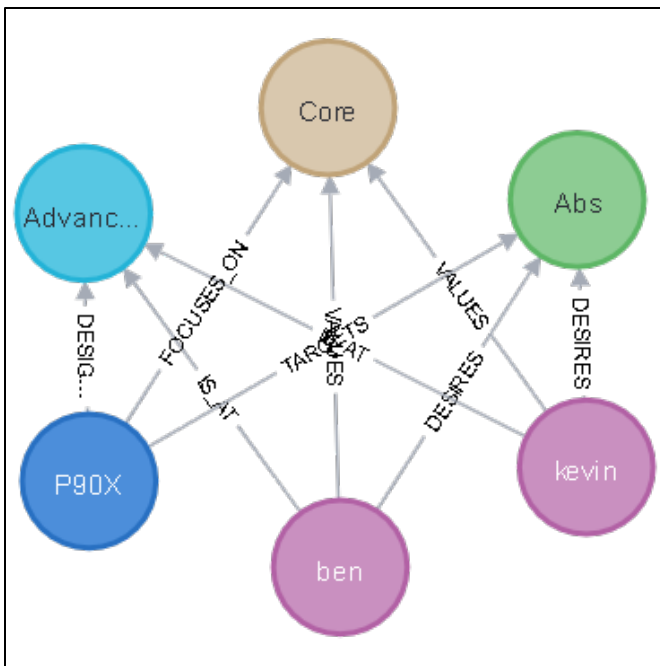
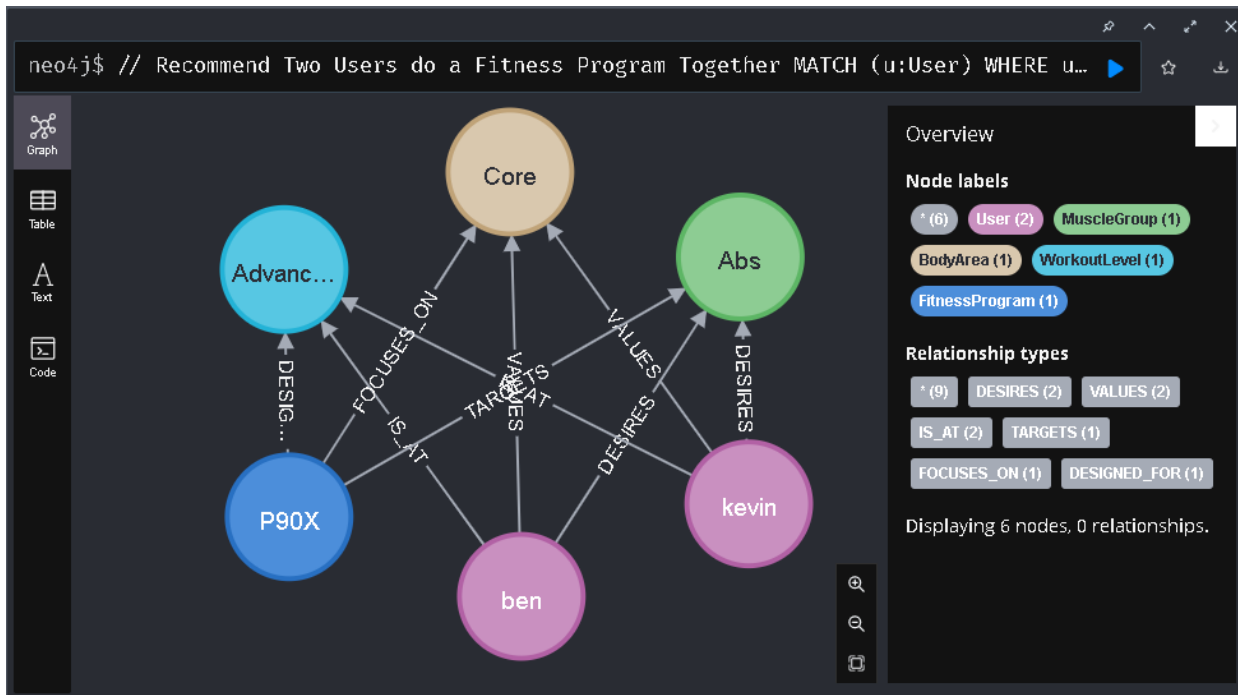
Recommend Two Users do a Fitness Program Together.

Here we make a few adjustments to recommend a FitnessProgram to a pair of Users that should experience it together.

Adjustments: Traverse out from the starting User's preferences to get other users connected to them as well, which makes the possible set of users to be recommended. Now we score, order and limit the users to the best one and then proceed to find the best fitness programs for those two users.

```
// Recommend Two Users do a Fitness Program Together
MATCH (u:User) WHERE u.username = "ben" OPTIONAL MATCH (u)-[:HAS]->(pl)
WITH u, pl MATCH (u)-[:IS_AT|PREFERS|DESIRES|VALUES]->(x)
WITH u, pl, x, coalesce(r.importance, 0.5) AS importance
MATCH (x)-[]-(x2) WHERE (x2:User) AND NOT (x2)-[:LIMITED_BY]->(pl) AND u <> x2
WITH u, pl, x2, collect({name: x.name, weight: importance}) AS traits
WITH u, pl, x2, reduce(s = 0, t IN traits | s + t.weight) AS score
WITH u, pl, x2, score ORDER BY score DESC LIMIT 1
WITH u, pl, x2, score OPTIONAL MATCH (x2)-[]->(x)-[]-(u), (x)-[:r]-
(fp:FitnessProgram) WHERE NOT (fp)-[:LIMITED_BY]->(pl)
WITH u, x2, score, collect(x) AS common, fp, collect({name: x.name, weight: coalesce(r.importance, 0.5)}) AS
traits
WITH u, x2, score, common, fp, reduce(s = 0, t IN traits | s + t.weight) AS score2
RETURN u, x2, score, common, fp, score2 ORDER BY score2 DESC LIMIT 1;
```

Graph Theory and Its Applications: Assignment-4



Conclusion

Upon a deep dive into the data of usage patterns of how users interact with workouts and supplements, we can make a clear inference that they are all very closely knit together.

Using Neo4j as a medium to visualize this data in a graph database does a great amount to recommend users with relevant workouts, supplements and even workout partners.

Recommendation systems of these kind play a key role in fitness apps and websites to deliver appropriate content to their users.

These systems can be further fine-tuned to fit the differing requirements of apps and websites.