

Homework Assignment #5

(Neo4J)

Your boss from the Couchbase assignment was once again very satisfied with your work, but went to a party last Friday where there was lots and lots of buzz about graph databases. All of their friends seem to be using them now. You have thus been asked to continue working at the startup, but to use Neo4J for the next set of tasks. In this assignment, you are to get a Neo4J instance working, load a graph version of the ShopALot data, and run some queries to get some new insights - focusing on connectivity insights - into the ShopALot data. (To save you potential pain, your boss hired an outside data wrangler to convert the previous ShopALot data dump into a Neo4J-ready form, so you now get to start from where they left off.)

Get Ready... (and set)

To start, follow the setup instructions **carefully** in order to install the **correct version** of Neo4J on your machine. In contrast to the previous assignments, you will now be loading a database dump file instead of the CSV or JSON formats you know and love, so the version matters a lot.

Go...!

It's time to get to work. Here's what you are to do using the Neo4J Browser app inside Neo4J Desktop, or by opening <http://localhost:7474> on your web browser:

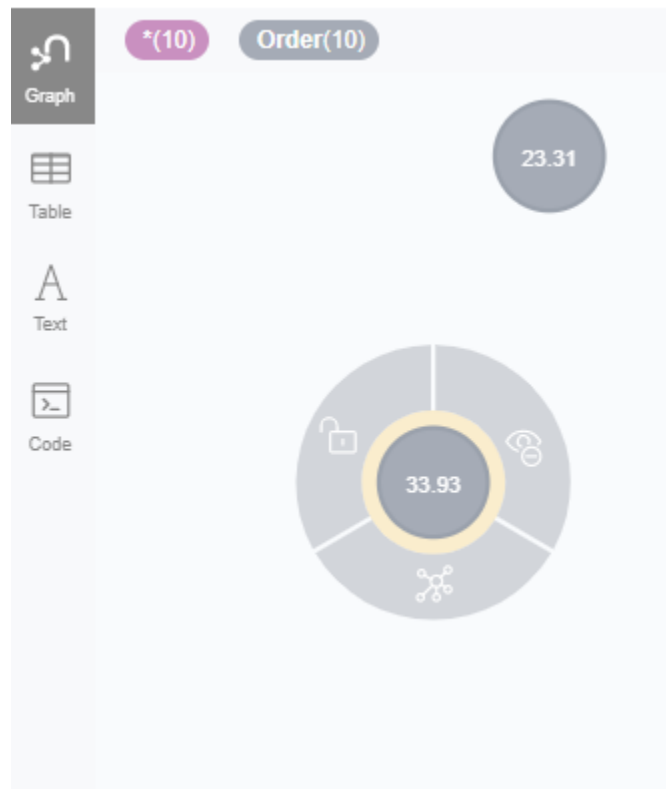
1. To get you familiar with the graph you have just loaded, run the commands below to answer questions A-C below.

```
CALL db.relationshipTypes;
CALL db.indexes;
CALL db.labels;
```

- A. What types of relationships exist in your graph?
 - B. What type(s) of indexes have we already built for you?
 - C. What are the labels in your graph?
2. Having a tighter grasp of what nodes and relationships exist in your graph, you now want to see how the two are intertwined (that is, which relationships are interconnecting which nodes). Luckily, Neo4J Desktop provides a great visual method for doing this: run the following query to get all nodes with the Order label:

```
MATCH (o:Order)
RETURN o
LIMIT 10;
```

From the result pane, click one of the nodes and press “Expand child relationships” (the button immediately below your node that looks like a tiny graph).



What types of relationships does an Order node have? What are the labels of the nodes that these relationships connect it to? Finally, what are the directions of each relationship?

3. Similar to other “schema-less” systems, two Neo4J nodes with the same label may have different properties and two relationships with the same type may each have different properties. Nodes aren’t even required to have labels! In Mongo Compass and the Couchbase UI, a schema-analysis tool was offered in their UIs in order to sample the documents and return the fields that a document in that collection might have. Neo4J desktop doesn’t offer as straightforward an approach to see what properties a given kind of node might have (the Database Information button, the disk icon in the top left hand corner, does however offer information about the DB as a whole), but you can resolve that by issuing a query to search through a subset of your nodes and extract the keys. An example for Customer nodes can be found below:

```
MATCH (c:Customer)
WITH c LIMIT 1000
UNWIND (keys(c)) AS custKeys
```

```
RETURN DISTINCT custKeys
```

Now for the question itself: Modify the query above and sample 1000 Vehicle nodes for their distinct properties.

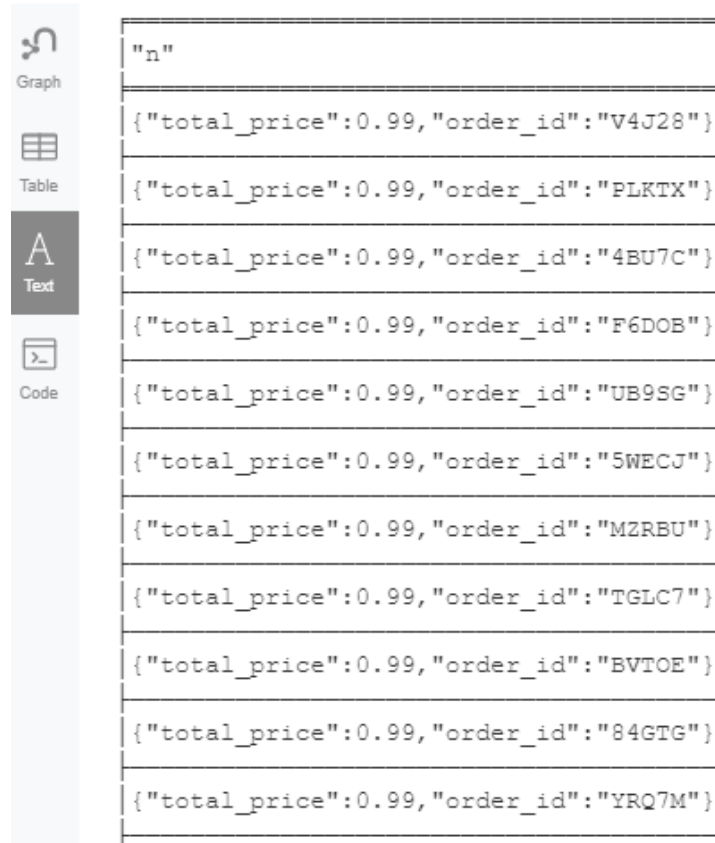
4. You are now ready to write some queries! While writing these Cypher queries, think about how you would formulate the equivalent SQL statement(s). Please answer the questions below:
 - A. To start your Cypher query writing journey, your boss has given you the following task: Write a Cypher query to return some OrderItem nodes (all properties). Sort the nodes by descending order of selling_price, and print just the first 10 nodes.
 - B. Cypher (and other graph query languages) must not only be able to search nodes based on their labels, but also based on nodes that they are connected to. Write a Cypher query to find paths to the Store node “Sheetz” that involve the FOR relationship. Return (any) two such paths.
 - C. You might have noticed that some customers haven’t placed any orders. With your curiosity piqued, you now want to find out how many of these customers exist in your graph. Write a Cypher query to count all the customers who have not placed an order.
 - D. In an effort to speed up access to product information, you may want to pre-fetch the products associated with an order if a user is currently reading an order in your application. Write a Cypher query to get all Product nodes associated with the order items contained in the order with order_id: “U7GWS”. Print the product names only.
 - E. Your company wants to target customers who are involved with not-so-popular stores, meaning customers who have placed orders for stores for which the total number of orders received is lower than a certain number. Write a Cypher query that returns the user ids of customers who have placed orders for stores that have received less than 10 orders. (Hint: use SIZE and a pattern involving Store and Order). Sort your results in ascending order and return the first 10 results.
 - F. A request has been issued from the head office: They want to determine exactly which shoppers are highly capable and have also fulfilled lots of orders. Write a Cypher query to return the a) user_ids, b) the capacity, and c) the number of orders fulfilled by shoppers with a capacity greater than 4 and an order count greater than 5. Sort your results in ascending order of user_id and return the first 10 results.
 - G. The future of ShopALot.com depends on having active users, and your boss wants to increase user activity by introducing a friends function to the website. To test the friends recommendation system, you would like to start by finding users who share an interest in certain stores. Write a Cypher query to return the distinct pairs of users of the customers who have bought at least two times from the store with store id “2TM62”. Each pair of nodes should be printed only once (e.g. show either (A, B) or (B, A) but not both!). Hint: each node has a unique id that can be retrieved by id(n) where n is the node. The ids can then be compared in the usual

- ways (<, <=, =, !=, >=, >). Print only the first names of both users, sorting by the ascending order of the first name of either user, and keep only the first 10 results.
- H. A “serve” path is defined as a path that starts from a shopper node, ends at a customer node, via an order node. We can say Shopper A serves Customer B if a serve path exists between them. One fact of the ShopALot world is that all of the shoppers are also customers. It is of particular interest whether two users (where both are shopper + customer) have served each other. (E.g., if A, as a shopper, has served B as a customer, while B, as a shopper, has also served A as a customer). Write a Cypher query to return all pairs of nodes where user A and user B have served each other. Again, (A, B) and (B, A) are considered to be duplications, so only one of the two pairs should be shown by your query.
 - I. To save the cost of identifying qualified pairs in problem H each time they are needed, your boss wants to add SERVE relationships between these users. Write a Cypher query that creates SERVE relationships between the users mentioned in problem H. The newly created SERVE relationships should go both ways for each pair of qualified users (i.e., A <-> B). Take a screenshot of the result of running your query; the result should start with “Created...”.
 - J. **[Extra Credit]** You have just received a new request from your boss: There is a customer (who is also a shopper) with user ID “SVT7J” who is curious about the potential dissemination of their information to other users. In particular, this customer is interested in knowing the length of the maximum shortest path from themselves to any other user via Order nodes. Write a Cypher query that will find the maximum (longest) shortest path length from this inquiring user to any other customer-shopper (a node with both Customer and Shopper labels). (Hint: take advantage of the shortestPath function.)
 - K. **[Extra Credit]** Continuing from the previous extra credit question, this same user (“SVT7J”) is now interested in knowing the actual user(s) involved in their shortest paths. Write another Cypher query to determine the user_id of user(s) whose shortest path is exactly 2 hops (i.e., 3 total nodes in the path) away from the user with user ID “SVT7J”.

What To Turn In

When you have finished your assignment, you should use Gradescope to turn in a PDF file that lists all queries you ran -- from start to finish -- and all the query results as they appear in Neo4j Browser’s “Text” format. Follow the steps below in order to generate your file for submission:

1. Download the “Homework #5 template.docx” from the course wiki.
2. Open the file. At the very top, specify (again) your name, student ID, and the date.
3. Fill in the Cypher query and query results to each question. To fill in the results, choose the “Text” format in the Neo4j Browser. You should see a format similar to:



"n"
{"total_price":0.99,"order_id":"V4J28"}
{"total_price":0.99,"order_id":"PLKTX"}
{"total_price":0.99,"order_id":"4BU7C"}
{"total_price":0.99,"order_id":"F6DOB"}
{"total_price":0.99,"order_id":"UB9SG"}
{"total_price":0.99,"order_id":"5WECJ"}
{"total_price":0.99,"order_id":"MZRBU"}
{"total_price":0.99,"order_id":"TGLC7"}
{"total_price":0.99,"order_id":"BVTOE"}
{"total_price":0.99,"order_id":"84GTG"}
{"total_price":0.99,"order_id":"YRQ7M"}

Take a screenshot of the table itself, including all of the borders. Insert the screenshot into the template file.

4. Export the template as PDF.
5. Finally, submit the PDF file to Gradescope!