**Appendix Q**

**Working with Neo4j**

|  |
| --- |
| **NOTE**  Most of this appendix, and all of the end of appendix problems, require the use of the Ch14\_FCC.txt file. The Appendix provides instructions on how to import this file into a Neo4j graph. |

**Answers to Review Questions**

**1. Explain the difference between using the same variable name and different variable names when matching multiple patterns in Neo4j.**

Within a given command, all references to a variable are treated as references to the same object (node, edge, or path). Therefore, if the same variable is used in multiple patterns in the same command, then the same node or edge will be required to match both patterns. If different variable names are used, then the node or edge does not have to be the same node or edge in both patterns.

**2. What is the difference between using WHERE and embedding properties in a node when creating a pattern in Neo4j?**

Embedded properties are much more limited. Embedded property specifications are treated as using an equality operator and combined using a logical AND. With a WHERE clause, other operators in addition to equality can be used such as less than, greater than, substrings, etc. Also, criteria in a WHERE clause can be combined with OR connectors as well as AND.

**Problem Solutions**

**For the following problems, use the Food Critics Club (FCC) graph database that was created and used earlier in the text for use with Neo4j.**

**1. Create a node that meets the following requirements. Use existing labels and property names as appropriate.**

**The node will be a member, and should be labeled as such, with member id 5000.**

**The member’s name is “Abraham Greenberg”.**

**Abraham was born in 1978, and lives in the state of “OH”.**

**Abraham’s email address is agreen@nomail.com, and his username is agberg.**

Create (:Member {mid:5000,

fname: "Abraham",

lname: "Greenberg",

birth: 1978,

state: "OH",

email: "agreen@nomail.com",

username: "agberg"

}

)

**2. Create a restaurant node with restaurant id is 10000, the name “Hungry Much”, and located in Cobb Place, KY.**

Create (:Restaurant {rid: 10000, name: "Hungry Much", state: "KY", city: "Cobb Place"})

**3. Update the “Hungry Much” restaurant created above to add the phone number “(931) 555-8888”, and a price rating of 2.**

Match(r :Restaurant {name:"Hungry Much"})

Set r.phone = "(931) 555-8888", r.price = 2

**4. Create a REVIEWED relationship between the member created above and the restaurant created above. The review should rate the restaurant as a 5 on taste, service, atmosphere, and value.**

Match (abe :Member {fname: "Abraham", lname: "Greenberg"}),

(hungry :Restaurant {name: "Hungry Much"})

Create (abe) -[rev :REVIEWED {taste: 5, service: 5, atmosphere: 5, value: 5}]-> (hungry)

**5. Create a REVIEWED relationship between member Frank Norwood and the restaurant created above. The review should rate the restaurant as a 4 on taste, service, and value, and rate the restaurant as a 2 on atmosphere.**

Match(frank :Member {fname: "Frank", lname: "Norwood"}), (hungry :Restaurant {name: "Hungry Much"})

Create(frank) -[rev :REVIEWED {taste:4, service: 4, atmosphere: 2, value: 4}]-> (hungry)

**6. Write a query to display member Frank Norwood and every restaurant that he has rated as a 4 or above on value.**

Match (frank :Member {fname: "Frank", lname: "Norwood"})

-[rev :REVIEWED]->

(rest :Restaurant)

Where rev.value >= 4

Return frank, rest

**7. Write a query to display cuisine, restaurant, and owner for every “American” or “Steakhouse” cuisine restaurant.**

Match (c :Cuisine) <-- (rest :Restaurant) <-[:OWNS]- (o :Owner)

Where c.name = "American" OR c.name = "Steakhouse"

Return c, rest, o

**8. Write a query to return the shortest path based only on reviews between members Abraham Greenberg and Herb Christopher.**

Match p = shortestPath((abe :Member {fname: "Abraham", lname: "Greenberg"})

-[:REVIEWED \*]-

(herb :Member {fname: "Herb", lname: "Christopher"}))

Return p