PantherID: 002571047 **Bhavin Patel**

Unstructured Data Management

CIS8045: In-Class Exercise 03

```
create (n1:CostCenter{id:1,Code:'CC1'});
create (n2:CostCenter{id:2,Code:'CC2'});
create (n3:Employee{id:3,Name:'Nathan',Surname:'Davies'});
create (n4:Employee{id:4,Name:'Rose',Surname:'Taylor'});
create (n5:Employee{id:5,Name:'Heather',Surname:'Underwood'});
create (n6:Employee{id:6,Name:'John',Surname:'Smith'});
create (n7:VacantPost{id:7});

match(from{id:3}),(to{id:1})create(from)-[r1:BELONGS_TO]->(to)set
r1.FROM='2011-01-10';

match(from{id:3}),(to{id:4})create(from)-[r2:REPORTS_TO]->(to);
match(from{id:4}),(to{id:1})create(from)-[r3:MANAGER_OF]->(to)set
r3.FROM='2010-02-08';
match(from{id:6}),(to{id:2})create(from)-[r4:REPORTS_TO]->(to)set
r4.FROM='2008-09-20';
match(from{id:6}),(to{id:5})create(from)-[r4:REPORTS_TO]->(to);
match(from{id:5}),(to{id:2})create(from)-[r4:BELONGS_TO]->(to);
match(from{id:5}),(to{id:7})create(from)-[r4:REPORTS_TO]->(to);
match(from{id:7}),(to{id:2})create(from)-[r4:BELONGS_TO]->(to);

//Find all employees, list name
match(e:Employee)
return e.Name, e.Surname

//Find all CC1 employees, list names
match(e:Employee)-[r]->(cc:CostCenter{Code:'CC1'})
return e.Name, e.Surname, r.FROM, cc.Code

match(e:Employee)-[r:BELONGS_TO]->(cc:CostCenter{Code:'CC1'})
return e.Name, e.Surname, r.FROM, cc.Code

//Find indirect links
match(e:Employee{Surname:'Smith'})-[*2]->(neighborhood)
return e.Name, e.Surname, neighborhood

//Range of hops
match(e:Employee{Surname:'Smith'})-[r*2..3]->(neighborhood)
return r, neighborhood
```

```
match(e:Employee{Surname:'Smith'})-[r*3..3]->(neighborhood)
return r, neighborhood

//Find all employees, list their possible reporting partners
match(e:Employee)
optional match (e)<-[r:REPORTS_TO]-(m:Employee)
return e.Surname, m.Surname

//Find a path
match p=(a{Surname:"Davies"})-[*]-(b{Surname:'Taylor'})
return p

//Find the shortest path
match p=(a{Surname:"Davies"})-[*]-(b{Surname:'Taylor'})
return allShortestPaths((a)-[*]-(b))

//recommendation
match(p:Person)-[f:FRIEND_OF]-()
return p,f
limit 50

match(p:Person)-[b:BOUGHT]-(prod:Product)
return p.name, count(prod)
order by count(prod) desc;

//Friendship based recommendation
match path=(p1:Person {name:'Lynda Willis'})-[:FRIEND_OF]- (p2:Person)-[:BOUGHT]-
>(prod:Product)
where not ((p1)-[:BOUGHT]-(prod))
return path

//Commonality based Recommendation
match (p1:Person)-[:HAS_COMPETENCY]-(c:Competency)-[:HAS_COMPETENCY]-
(p2:Person)
with p1,p2
match(p1)-->(co:Company)<--(p2)-[:BOUGHT]->(prod:Product)
where not ((p1)-[:BOUGHT]-(prod))
return p1.name, p2.name, prod.name

//Friendship Gap
match path=(p1:Person)-[:FRIEND_OF*2..2]-(p2:Person)
where not ((p1)-[:FRIEND_OF]-(p2))
return path
limit 10
```

```
//Friendship recommendation
match path=(p1:Person)-[:HAS_COMPETENCY]-(c:Competency)-
[:HAS_COMPETENCY]-(p2:Person)
with path, p1,p2
match(p1)-->(co:Company)<--(p2)
where not ((p1)-[:FRIEND_OF]-(p2))
return path
limit 10

//Degree Centrality
match(n:Person)-[r:FRIEND_OF]->(m:Person)
return n.name, count(r) as DegreeCentrality
order by DegreeCentrality desc;

//Betweenness Centrality
match p=allShortestPaths((n:Person)-[:FRIEND_OF*]-(m:Person))
where id(n) < id(m) and length(p) > 1
unwind nodes(p)[1..-1] as allnodes
return allnodes.name, count(*) as BetweennessCentrality
order by BetweennessCentrality

//Trump World
//Most connected organizations
match (o:Organization)-[r]-()
return o.name, count(*), collect(distinct type(r)) as types
order by count(*) desc limit 10

//The second degree network of Kushner
match network=(:Person{name:"JARED KUSHER"})-[*..2]-()
return network

//Trump's possible connections to Putin
match path=allShortestPaths((dt:Person{name:"DONALD J. TRUMP"})-[*]-
(vp:Person{name:"VLADIMIR PUTIN"}))
return path

//Clustering Coefficient:
match path1=(p0:Person)-[:Related_To] - (dt:Person{name:"DONALD J. TRUMP"})-
[:Related_To]-(p2:Person)
where not ((p0)-[:Related_To]-(p2)) and id(p0)<id(p2)
with count(path1) as m

match path2=(p0:Person)-[:Related_To] - (dt:Person{name:"DONALD J. TRUMP"})-
[:Related_To]-(p2:Person)
where id(p0)<id(p2)
return 1.0*m/count(path2) as Cluster_coeff
```