Neo4j Practicals

Song Database

1. Consider a Song database, with labels as Artists, Song,Recording_company, Recoding_studio, song author etc. Relationships can be as follows

Artist→[Performs]→Song→[Written by]→Song_author. Song→[Recorded in]→Recording Studio→[managed by]→recordingCompany Recording Company→[Finances] →Song You may add more labels and relationship and their properties, as per assumptions.

Create Database

Node

A. create(person:Artist{name:'Atif Aslam',address:'Mumbai',age:'30'})
B. create(Naina:Song{name:'Naina Re',release:'2015',viewers:'100M'})
C. create(person:Song_Author{name:'sameer Anjan',address:'Mumbai',age:'40'})
D. create(place:Studio{name:'Source',address:'Mumbai'})
E. create(company:RecordingCompany{name:'OFF Keys',address:'Mumbai'})

Relationship

A. match(a:Song),(b:Artist) where a.name="Naina Re" and b.name="Atif Aslam" create(b)-[:performs]->(a)
B. match(a:Song_Author),(b:Song) where a.name='sameer anjan' and b.name='Naina Re' create(b)-[:Written_by]->(a)
C. match(a:Song),(b:Studio) where a.name="Naina Re" and b.name="Source" create(a)-[:Recordedin]->(b)
D. match(a:RecordingCompany),(b:Studio) where a.name="OFF Keys" and b.name="Source" create(b)-[:manageby]->(a)
E. match(a:RecordingCompany),(b:Song) where a.name="OFF Keys" and b.name="Naina Re" create(a)-[:finances]->(b)

Query

a) List the names of songs written by ":.....".

```
Match(s:Song)-[r:Written_by]->(a:Song_Author) where a.name='sameer Anjan' return s.name
```

b) List the names of record companies who have financed for the song "...."

```
Match(rc:RecordingCompany)-[r:finances]->(s:Song) where s.name='Naina Re' return rc.name
```

c) List the names of artist performing the song "....."

```
Match(a:Artist)-[r:performs]->(s:Song) where s.name='Naina Re' return a.name
```

d) Name the songs recorded by the studio " ...."

```
Match(a:Song)-[r:Recordedin]->(s:Studio) where s.name='Source' return a.name
```

e) List the names of artists who have sung only songs written by "……."

```
Match(a:Artist)-[r:performs]->(s:Song)-[w:Written_by]->
(sa:Song_Author) where sa.name='sameer Anjan' return a.name
```

f) List the names of artists who have sung the maximum number of songs recorded by "……" studio

```
Match(a:Studio{name:'Source'})<-[r:Recordedin]-(s:Song)<-[p:performs]-
>(b:Artist) with b,count(s) as songCount order by songCount DESC limit 1 return b.name
```

g) List the names of songs financed by "…..", and sung by "……."

```
Match(a:Artist)-[r:performs]->(s:Song)<-[f:fianances]-
>(rc:RecordingCompany) where a.name='Atif Aslam' and rc.name='OFF Keys' return s.name
```

Employee Database

Consider an Employee database, with a minimal set of labels as follows

Employee: denotes a person as an employee of the organization Department: denotes the different departments, in which employees work. Skillset: A list of skills acquired by an employee Projects: A list of projects in which an employee works.

A minimal set of relationships can be as follows: Works_in : employee works in a department Has_acquired: employee has acquired a skill Assigned_to : employee assigned to a project Controlled_by: A project is controlled by a department Project_manager : Employee is a project_manager of a Project

Create database

Nodes

A. `create(person:Emp{name:'ABC',address:'Mumbai'})`
B. `create(project:Project{name:'XYZ',type:'Website'})`
C. `create(dept:Department{name:'PQR',type:'Development'})`
D. `create(skill:SkillSet{name:'FrondEnd Developement',type:'Developer'})`

Relationship

A. `match(a:Emp),(b:Department) where a.name="ABC" and b.name="PQR" create (a)-[:Worksin]->(b)`
B. `match(a:Emp),(b:SkillSet) where a.name="ABC" and b.name="FrondEnd Developement " create (a)-[:Has_aquired]->(b)`
C. `match(a:Department),(b:Project) where a.name="PQR" and b.name="XYZ" create (a)<-[:Controlled_by]-(b)`
D. `match(a:Emp),(b:Project) where a.name="ABC" and b.name="XYZ" create (a)-[:Assigned_to]->(b)`

Query

a) List the names of employees in department "……."

```
match(a:Emp)-[r:Worksin]->(b:Department) where b.name="PQR" return a.name
```

b) List the projects along with their properties, controlled by department "……"

```
match(a:Project)-[r:Controlled_by]->(b:Department) where b.name="PQR" return a
```

c) List the departments along with the count of employees in it

```
match(a:Emp)-[r:Worksin]-
>(b:Department) with b,count(a) as empCount return b.name,empCount
```

d) List the skillset for an employee "…….

```
match(a:Emp)-[r:Has_aquired]->(b:SkillSet) where a.name="ABC" return b.name
```

e) List the names of employees having the same skills as employee "……."

```
match(a:Emp{name:"ABC"})-[r:Has_aquired]->(b:SkillSet) match(c:Emp)-[s:Has_aquired]-
>(b) where c.name="DEF" return a.name
```

f) List the projects controlled by a department "……." and have employees of the same department working in it.

```
match(a:Emp)-[r:Worksin]->(b:Department)<-[c:Controlled_by]-
(d:Project) where b.name="PQR" return d.name
```

g) List the names of the projects belonging to departments managed by employee "……."

```
match(a:Emp)-[r:Worksin]->(b:Department)<-[c:Controlled_by]-
(d:Project) where a.name="ABC" return d.name
```

Social Network Database

Create a Social network database , with labels as Person, Affiliations, Groups, Story, Timeline etc. Some of the relationships can be as follows:

Person→[friend of]→Person→[affiliated to]→affiliations Person→[belongs to]→Groups, Person→[create]→Story→[refers to]→Person Person→[creates]→Timeline→[reference for]→Story , Timeline→[contains]→Messages

Nodes

A. `create(john:Person{name:'John',address:'Pune',birthyear:'1995'})`
B. `create(friend:Person{name:'Tom',address:'Pune',birthyear:'1995',sinceyear:'2015'})`

C. create(a:Affiliation{name:'Society',address:'Pune'})
D. create(g:Group{name:'Group1',address:'Pune'})
E. create(s:Story{name:'Tom & Jerry',release:'2020'})
F. create(t:Timeline{start:'2017',end:'2020'})
G. create(m:Message{text:'Complete the story in time'})

Relationship

A. match (a:Person),(b:Person) where a.name='Tom' and b.name='John' create (a)-[r:friends_of]->(b)
B. match (a:Person),(b:Affiliation) where a.name='John' and b.name='Society' create (a)-[r:affiliated_to]->(b)
C. match (a:Person),(b:Group) where a.name='John' and b.name='Group1' create (a)-[r:belongs_to]->(b)
D. match (a:Person),(b:Story) where a.name='John' and b.name='Tom & Jerry' create (a)-[r:creates]->(b)
E. match (a:Person),(b:Story) where a.name='Tom' and b.name='Tom & Jerry' create (a)<-[r:refers_to]-(b)
F. match (a:Person),(b:Timeline) where a.name='John' and b.start='2017' create (a)-[r:create]->(b)
G. match (a:Timeline),(b:Story) where a.start='2017' and b.name='Tom & Jerry' create (a)-[r:reference_for]->(b)
H. match (a:Timeline),(b:Message) where a.start='2017' and b.text='Complete the story in time' create (a)-[r:contains]->(b)

Query

a) Find all friends of “John”, along with the year, since when john knows them.

```
match (a:Person)-[r:friends_of]-
>(b:Person) where b.name="John" and a.sinceyear='2015' return a.name
```

b) List out the affiliations of John.

```
match (a:Person)-[r:affiliated_to]->(b:Affiliation) where a.name="John" return b.name
```

c) Find all friends of john, who are born in the same year as John

```
match (a:Person)-[r:friends_of]-
>(b:Person) where b.name="John" and a.birthyear=b.birthyear return a.name
```

d) List out the messages posted by John in his timeline, during the year 2015.

```
match (a:Person)-[r:create]->(b:Timeline)-[s:contains]-
>(c:Message) where a.name="John" and b.start='2017' return c.text
```

e) List out the people, who have created maximum timeline messages.

```
match(a:Person)-[r:create]->(b:Timeline)-[s:contains]-
>(c:Message) with a,count(c) as msgCount order by msgCount DESC limit 1 return a.name,msg
Count
```

f) List all friends of John's friend, Tom

```
match (a:Person{name:'John'})<-[r:friends_of]-(b:Person)<-[s:friends_of]-
(c:Person) where c<>a return c.name
```

g) List the people with maximum friends

```
match (a:Person)-[r:friends_of]-
>(b:Person) with b,count(a) as fCount order by fCount DESC limit 1 return b.name,fCount
```

h) List the people who are part of more than 3 group

```
match (a:Person)-[r:belongs_to]-
>(b:Group) with a,count(b) as gCount where gCount>3 return a.name,gCount
```

Movie Database

Consider a movie database, with nodes as Actors, Movies, Roles, Producer, Financier, Director. Assume appropriate relationships between the nodes, include properties for nodes and relationships.

Node

A. `create(ddlj:Movie{name:'DDLJ',release:'1990'})`
B. `create(kkhh:Movie{name:'Dilwale',release:'2018'})`
C. `create(kajol:Actor{name:'Kajol'})`
D. `create(sk:Actor{name:'Shahrukh Khan'})`
E. `create(r:Role{Type:'Lead'})`
F. `create(r:Producer{name:'Rohit Shetty'})`
G. `create(r:Director{name:'Rohit Shetty'})`
H. `create(r:Reviewer{name:'ABC'})`
I. `create(r:Reviewer{name:'PQR'})`

Relationship

A. `match(m:Movie),(a:Actor) where m.name='Dilwale' and a.name='Kajol' create(a)-[:Actedin]->(m)`
B. `match(m:Movie),(a:Actor) where m.name='DDLJ' and a.name='Shahrukh Khan' create(a)-[:Actedin]->(m)`
C. `match(m:Movie),(a:Actor) where m.name='Dilwale' and a.name='Shahrukh Khan' create(a)-[:Actedin]->(m)`
D. `match(m:Movie),(a:Director) where m.name='Dilwale' and a.name='Rohit Shetty' create(m)-[:Directedby]->(a)`

E. match(m:Movie),(a:Producer) where m.name='DDLJ' and a.name='Rohit Shetty' create(m)-[:Producedby]->(a)
F. match(m:Movie),(a:Reviewer) where m.name='Dilwale' and a.name='ABC' create(m)-[:Reviewedby]->(a)
G. match(m:Reviewer),(a:Reviewer) where m.name='ABC' and a.name='PQR' create(m)-[:Followedby]->(a)
H. match(m:Reviewer),(a:Reviewer) where m.name='ABC' and a.name='PQR' create(a)-[:Followedby]->(m)
I. match(a:Actor),(r:Role) where a.name='Kajol' and r.type='Lead' create (a)-[:Played]->(r)

Query

a) Find all actors who have acted in a movie "……."

```
match(a:Actor)-[r:Actedin]->(m:Movie) where m.name='DDLJ' return a.name
```

b) Find all reviewer pairs, one following the other and both reviewing the same movie, and return entire subgraphs.

```
match(a:Reviewer)-[r:Followedby]->(b:Reviewer)<-[s:Reviewedby]-
(m:Movie) where m.name='Dilwale' return a,b,m
```

c) Find all actors that acted in a movie together after 2010 and return the actor names and movie node

```
match(a:Actor)-[r:Actedin]->(m:Movie) where m.release>2010 WITH a, COLLECT(DISTINCT m) AS movies
WHERE SIZE(movies) > 1 return a.name,movies
```

d) Find all movies produced by " ……."

```
match(a:Producer)<-[r:Producedby]-(m:Movie) where a.name='Rohit Shetty' return m.name
```

e) List the names of actors that paired in multiple movies together.

```
match(a:Actor)-[r:Actedin]-
>(m:Movie) with a,count(m) as mCount where mCount>1 return a.name
```

f) List all pairs of actor–movie subgraphs along with the roles played.

```
match(s:Role)<-[t:Played]-(a:Actor)-[r:Actedin]->(m:Movie) where s.type='Lead' return a,m,s
```

g) List all reviewers and the ones they are following directly or via another a third Reviewer

```
match(a:Reviewer)-[r:Followedby]->(b:Reviewer) where a.name='ABC' return a,b
```

h) List the names of movies that have the most number of review

```
match(a:Movie)-[r:Reviewedby]-
>(b:Reviewer) with a,count(b) as review order by review DESC limit 1 return a.name,review
```

Library Database

Node

A. create (a:Book{title:'Ninteen Eighty Four',status:'issued',condition:'new',cost:'350',type:'novel'})
B. create (a:Book{title:'Brave New World', status:'issued', condition:'new',cost:'400',type:'novel'})
C. create (a:Author{name:'George Orwell',born:'25/6/1903',died:'21/1/1950', city:'Mumbai'})
D. create (a:Author{name:'Aldous Huxley',born:'26/7/1894',died:'22/11/1963', city:'pune'})
E. create (a:Reader{name:'Ram',born:'25/1/1997'})
F. create (a:Publisher{name:'Windus',city:'pune'})
G. create (a:Publisher{name:'Secker',city:'pune'})

Relationship

A. match(a:Book),(b:Author) where a.title='Ninteen Eighty Four' and b.name='George Orwell' create(b)-[:Wrote]->(a)
B. match(a:Book),(b:Author) where a.title='Brave New World' and b.name='Aldous Huxley' create(b)-[:Wrote]->(a)
C. match(a:Book),(b:Reader) where a.title='Brave New World' and b.name='Ram' create(a)-[:Issuedby]->(b)
D. match(a:Book),(b:Reader) where a.title='Brave New World' and b.name='Ram' create(b)-[:Recommended]->(a)
E. match(a:Book),(b:Publisher) where a.title='Brave New World' and b.name='Windus' create(a)-[:Publishedby]->(b)
F. match(a:Book),(b:Publisher) where a.title='Ninteen Eighty Four' and b.name='Secker' create(a)-[:Publishedby]->(b)

Query

a) List all people, who have issued a book "……"

```
match(a:Book)-[r:Issuedby]->(b:Reader) where a.title='Brave New World' return b.name
```

b) Count the number of people who have read " …."

```
match(a:Book)-[r:Issuedby]->(b:Reader) where a.title=Ninteen Eighty Four' return b.name
```

c) Add a property "Number of books issued " for Mr. Joshi and set its value as the count

```
match(a:Reader{name:'Mr.Joshi'}) with a match(a)<-[r:Issuedby]-
(b:Book) with a,count(b) as bCount set a.numberofbooksIssued=bCount
```

d) List the names of publishers from pune city.

```
match(a:Publisher) where a.city='pune' return a.name
```

e) List all readers who have recommended either book "…" or "……." or "……."

```
match(a:Book)<-[r:Recommended]-
(b:Reader) where a.title='Brave New World' or a.title='Ninteen Eighty Four' or a.title='Jane
Eyre' return b.name
```

f) List the readers who haven't recommended any book

```
match(a:Book)<-[r:Recommended]-
(b:Reader) with b,count(a) as rCount where rCount<1 return b.name,rCount
```

g) List the authors who have written a book that has been read / issued by maximum number of readers.

```
match(c:Author)-[s:Wrote]->(a:Book)-[r:Issuedby]-
>(b:Reader) with c,count(b) as reader order by reader DESC limit 1 return c.name,reader
```

h) List the names of books recommended by "……." And read by at least one reader

```
match(c:Reader)-[s:Recommended]->(a:Book)-[r:Issuedby]-
>(b:Reader) with c,a,count(b) as rCount where c.name='Ram' and rCount>=1 return a.title,rCoun
t
```

i) List the names of books recommended by "……" and read by maximum number of readers.

```
match(c:Reader)-[s:Recommended]->(a:Book)-[r:Issuedby]-
>(b:Reader) with a,c,count(b) as rCount order by rCount DESC limit 1 where c.name='Ram' retu
rn a.title,rCount
```

j) List the names of publishers who haven't published any books written by authors from Pune and Mumbai.

```
match(a:Author)-[r:Wrote]->(b:Book)-[s:Publishedby]-
>(p:Publisher) where a.city<>'Mumbai' and  a.city<>'Pune' return p.name
```

k) List the names of voracious readers in our library

```
match(a:Book)-[r:Issuedby]-
>(b:Reader) with b,count(a) as rCount order by rCount DESC limit 1 return b.name,rCount
```