Model the following Society relations among people working in “HCL”, as a graph model, and answer the queries using Cypher. A person can be a friend of another person. A person may have siblings (brothers / sisters), A person may be a parent(mother/father) of another person. A person stays either in Pune or Mumbai or Kolhapur. A person may be working on either ‘Finance’ or ‘Inventory’ or ‘Sales’ projects.

1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model. [10]

2. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [5]

3. Answer the following Queries in Cypher:

a. List the names of people who are parents. [2 1/2]

b. List the names of people working on ‘Finance ‘project [2 1/2]

c. List the names of people staying in ‘Pune’ and ‘Mumbai’. [5]

d. List the names of people who are mothers.

#DipaliMeher

[5] [Note: You may Assume additional labels and properties depending on the query requirements]

create(p:Person{name:"Radhika",age:69}) return p

create(p:Person{name:"Dipali",age:39}) return p

create(p:Person{name:"Pallawi",age:43}) return p

create(p:Person{name:"Meenal",age:38}) return p

create(p:Person{name:"Prakash",age:40}) return p

create(p:Person{name:"Mrunu",age:2}) return p

create(p:Person{name:"Rugved",age:4}) return p

create(p:Person{name:"Shreya",age:2}) return p

create(p:Person{name:"Omkar",age:4}) return p

create(p:Person{name:"Kaustubh",age:45}) return p

A person can be friend of another person

match(p:Person),(pp:Person) where p.name="Dipali" and pp.name="Pallawi" create(p)-[:Friend\_of]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Dipali" and pp.name="Meenal" create(p)-[:Friend\_of]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Mrunu" and pp.name="Rugved" create(p)-[:sisterof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Omkar" and pp.name="Shreya" create(p)-[:brotherof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Meenal" and pp.name="Mrunu" create(p)-[:motherof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Meenal" and pp.name="Rugved" create(p)-[:motherof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Kaustubh" and pp.name="Mrunu" create(p)-[:fatherof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Kaustubh" and pp.name="Rugved" create(p)-[:fatherof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Dipali" and pp.name="Shreya" create(p)-[:motherof]->(pp)return p,pp

match(p:Person),(pp:Person) where p.name="Prakash" and pp.name="Shreya" create(p)-[:fatherof]->(pp)return p,pp

#DipaliMeher

match(p:Person),(pp:Person) where p.name="Radhika" and pp.name="Rrugved" create(p)-[:grandmotherof]->(pp)return p,pp

person stays in pune Mumbai or Kolhapur

create(c:City{name:"Pune"}) return c

create(c:City{name:"Mumbai"}) return c

create(c:City{name:"Kolhapur"}) return c

match(p:Person),(c:City) where p.name="Prakash" and c.name="Pune" create(p)-[:staysin]->(c)return p,c

match(p:Person),(c:City) where p.name="Kaustubh" and c.name="Mumbai" create(p)-[:staysin]->(c)return p,c

match(p:Person),(c:City) where p.name="Meenal" and c.name="Mumbai" create(p)-[:staysin]->(c)return p,c

match(p:Person),(c:City) where p.name="Dipali" and c.name="Pune" create(p)-[:staysin]->(c)return p,c

create(pr:Project{name:"Finance"}) return pr

create(pr:Project{name:"Inventory"}) return pr

#DipaliMeher

create(pr:Project{name:"Sales"}) return pr

match(p:Person),(pr:Project) where p.name="Prakash" and pr.name="Finance" create(p)-[:workson]->(pr)return p,pr

match(p:Person),(pr:Project) where p.name="Kaustubh" and pr.name="Inventory" create(p)-[:workson]->(pr)return p,pr

match(p:Person),(pr:Project) where p.name="Dipali" and pr.name="Sales" create(p)-[:workson]->(pr)return p,pr

Queries

1) List the names of people who are parents

🡪MATCH (p:Person),(pp:Person) Where

(p)-[:fatherof]->(pp)

RETURN DISTINCT p.name AS  name1

UNION ALL

 MATCH (p:Person),(pp:Person) Where

(p)-[:motherof]->(pp) RETURN DISTINCT  p.name AS name1

OR

MATCH (p:Person),(pp:Person) Where

(p)-[:fatherof]->(pp) or (p)-[:motherof]->(pp)

RETURN DISTINCT p.name

2) List the names of people working on ‘Finance ‘project [2 1/2]

🡪 match(p:Person),(pr:Project) where pr.name="Finance" and(p)-[:workson]->(pr)

return p.name

3) List the names of people staying in ‘Pune’ and ‘Mumbai’. [5]

🡪 MATCH (p:Person),(c:City) WHERE c.name IN ['Pune', 'Mumbai'] and (p)-[:staysin]->(c) RETURN p.name

4) List the names of people who are mothers. [5]

🡪 match(p:Person),(pp:Person)  where  (p)-[:motherof]->(pp) return DISTINCT  p.name

5) Display the names of people living in Mumbai.

#DipaliMeher

🡪 match(p:Person),(c:City) where c.name="Mumbai" and (p)-[:staysin]->(c) return p.name

6) Display the nodes having age above 40.

🡪 match(p:Person) where p.age>40 return p.name

7)List the names of people who are grandmothers.

🡪MATCH (p:Person),(pp:Person) Where

(p)-[:grandmotherof]->(pp)

RETURN DISTINCT p.name AS Grandmother

8)List the names of people having the most friends.

query to find friends of each person

MATCH (p:Person)-[:Friend\_of]->(friend:Person) RETURN p.name, collect(friend.name) AS friend

query to find number of friends of eachperson

MATCH (p:Person)-[:Friend\_of]->(friend:Person) RETURN p.name, size(collect(friend.name)) AS numberOfFriends;

MATCH (p:Person)-[:Friend\_of]->(pp:Person) with p.name as names, count(pp.name) as count\_vt return names,count\_vt

MATCH (p:Person)-[:Friend\_of]->(pp:Person) with p.name as names, count(pp.name) as count\_vt with collect{names:names,count\_vt:count\_vt}as rows, max(count\_vt) as max UNWIND[row in rows WHERE row.count\_vt=max] as row  RETURN row.names as names,row.count\_vt as count\_vt

MATCH (p:People)-[:recommands]->(c:Course) with c.name as names, count(c.name) as count\_vt WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

MATCH (p:Person)-[r:Friend\_of]->(pp:Person) with pp.name as names , count(pp.name) as count\_vt

MATCH (p:People)-[:recommands]->(c:Course) with c.name as names, count(c.name) as count\_vt WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

MATCH (p:Person)-[r:Friend\_of]->(friend:Person) RETURN p.name,count(r) size(collect({nm1:friend.name,count\_vt:r) AS numberOfFriends

WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

#DipaliMeher

🡪match(p:Person)-[:Friend\_of]->(pp:Person) return count(\*)

MATCH (u:User {lastName:"Example"})-[:PINNED]->(z:Pin) WITH collect(z) as cs MATCH (:User)-[:LIKES]->(y:Pin) WHERE (y) in cs MATCH (a:User)-[r:LIKES]->(y) RETURN COUNT(r), y

MATCH (p:Person)-[r:Friend\_of]->(pp:Person) WITH collect(pp) as cs MATCH (p)-[:LIKES]->(y:Pin) WHERE (y) in cs MATCH (a:User)-[r:LIKES]->(y) RETURN COUNT(r), y

MATCH (p:Person)-[:Friend\_of]->(pp:Person),(d:Department)-[:Conducts]->(c:Course) where d.name="Zoology" with c.name as names, count(c.name) as count\_vt WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

Model the following University information system as a graph model, and answer the following queries using Cypher. University has various departments like Mathematics, Geology, Chemistry, etc. Each department conducts various courses and a course may be conducted by multiple departments. Every course may have recommendations provided by people.

1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model. [5]

2. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [10]

3. Answer the Queries a. List the names of the courses provided by Chemistry Department. [21/2]

b. List the details of all the departments in the university. [21/2]

c. List the names of common courses across chemistry and zoology department. [5]

d. List the most recommended course in Computer Science Department. [5]

[Note: You may Assume additional labels and properties depending on the query requirements]

create(:University{name:'SPPU',location:'Pune'})

Department Creation

create (:Department{name:'Comp-Sci'})

create (:Department{name:'Zoology'})

create (:Department{name:'Maths'})

create (:Department{name:'Chemistry'})

University Department Relationship

match(u:University),(d:Department) where u.name='SPPU'and d.name='Zoology' create (u)-[:Has]->(d) return u,d

match(u:University),(d:Department) where u.name='SPPU'and d.name='Chemistry' create (u)-[:Has]->(d) return u,d

match(u:University),(d:Department) where u.name='SPPU'and d.name='Maths' create (u)-[:Has]->(d) return u,d

match(u:University),(d:Department) where u.name='SPPU'and d.name='Comp-Sci' create (u)-[:Has]->(d) return u,d

Course Creation

create (:Course{name:'ExtraCredit'})

create (:Course{name:'BBA'})

create (:Course{name:'Analytical'})

#DipaliMeher

create (:Course{name:'Fish culture'})

department and course matching

match(c:Course),(d:Department) where d.name='Comp-Sci' and c.name='BBA' create (d)-[:Conducts]->(c) return d,c

match(c:Course),(d:Department) where d.name='Maths' and c.name='BBA' create (d)-[:Conducts]->(c) return d,c

match(c:Course),(d:Department) where d.name='Maths' and c.name='ExtraCredit' create (d)-[:Conducts]->(c) return d,c

match(c:Course),(d:Department) where d.name='Zoology' and c.name='ExtraCredit' create (d)-[:Conducts]->(c) return d,c

match(c:Course),(d:Department) where d.name='Chemistry' and c.name='ExtraCredit' create (d)-[:Conducts]->(c) return d,c

people creation

create (:People{name:'Dipali'})

create (:People{name:'Maharaj'})

create (:People{name:'Datta'})

people give recommandations to course

MATCH (p:People), (c:Course) WHERE p.name = "Datta" AND c.name = "ExtraCredit" CREATE (p)-[:recommands ]->(c) RETURN p,c

MATCH (p:People), (c:Course) WHERE p.name = "Maharaj" AND c.name = "ExtraCredit" CREATE (p)-[:recommands ]->(c) RETURN p,c

MATCH (p:People), (c:Course) WHERE p.name = "Dipali" AND c.name = "ExtraCredit" CREATE (p)-[:recommands ]->(c) RETURN p,c

MATCH (p:People), (c:Course) WHERE p.name = "Dipali" AND c.name = "BBA" CREATE (p)-[:recommands ]->(c) RETURN p,c

Query 1) List the details of all the departments in the university

match(u:University),(c:Course),(d:Department) where u.name='SPPU' and (d)-[:Conducts]->(c) and (u)-[:Has]->(d)return d.name,c.name

Query 2) List the names of common courses across computer Science and Maths department.

 WITH ['Comp-Sci','Maths'] as names MATCH (d:Department) WHERE d.name in names WITH collect(d) as s MATCH (c:Course) WHERE ALL(d in s WHERE (d)-[:Conducts]->(c)) RETURN c

#DipaliMeher

Query 3) courses run by chemistry department

match(c:Course),(d:Department) where d.name='Chemistry' and (d)-[:Conducts]->(c) return d,c,c.name

Query 4) List the most recommended course in Zoology Department.

MATCH (p:People)-[:recommands]->(c:Course),(d:Department)-[:Conducts]->(c:Course) where d.name="Zoology" with c.name as names, count(c.name) as count\_vt WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

Model the following Furniture Showroom information as a graph model, and answer the queries using Cypher. Consider a furniture showroom with different types of furniture like sofas sets, tea tables, cupboards, beds, dining tables, etc. Showroom is divided into different sections, one section for each furniture type, each section is handled by a sales staff. A sales staff can handle one or more sections. Customers may enquire about furniture. An enquiry may result in a purchase by the customer.

1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model. [5]

#DipaliMeher

2. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [10]

3. Answer the queries.

a. List the types of furniture available in the showroom. [21/2]

b. List the sections handled by Mr. Satish. [21/2]

c. List the names of customers who have done only enquiry but not made any purchase. [5]

d. List the fast-moving furniture types. [5]

[Note: You may Assume additional labels and properties depending on the query requirements]

create(s:Section{name:"Sofa"}) return s

create(s:Section{name:"Table"}) return s

create(s:Section{name:"Bed"}) return s

create(s:Section{name:"Cupboard"}) return s

create(f:Furniture{name:"Coffee Table",color:"White",price:2000}) return f

create(f:Furniture{name:"Computer Table",color:"Red",price:2450}) return f

create(f:Furniture{name:"L-Shape Sofa",color:"Brown",price:200000}) return f

create(f:Furniture{name:"Sofa cum bed",color:"Green",price:240000}) return f

create(f:Furniture{name:"Steel cupboard",color:"black",price:40000}) return f

create(f:Furniture{name:"bed with storage",color:"black",price:40000}) return f

create(f:Furniture{name:"bed without storage",color:"black",price:40000}) return f

match(f:Furniture),(s:Section) where f.name="Computer Table" and s.name="Table" create(f)-[:belongsto]->(s) return f,s

match(f:Furniture),(s:Section) where f.name="Coffee Table" and s.name="Table" create(f)-[:belongsto]->(s) return f,s

#DipaliMeher

match(f:Furniture),(s:Section) where f.name="L-Shape Sofa" and s.name="Sofa" create(f)-[:belongsto]->(s) return f,s

match(f:Furniture),(s:Section) where f.name="Sofa cum bed" and s.name="Sofa" create(f)-[:belongsto]->(s) return f,s

match(f:Furniture),(s:Section) where f.name="Steel cupboard" and s.name="Cupboard" create(f)-[:belongsto]->(s) return f,s

match(f:Furniture),(s:Section) where f.name="bed with storage" and s.name="Bed" create(f)-[:belongsto]->(s) return f,s

match(f:Furniture),(s:Section) where f.name="bed without storage" and s.name="Bed" create(f)-[:belongsto]->(s) return f,s

create(ss:Staff{name:"Satish"})return ss

create(ss:Staff{name:"Meenal"})return ss

create(ss:Staff{name:"Dipali"})return ss

match(s:Section),(ss:Staff) where s.name="Table" and ss.name="Satish" create(s)-[:handaledby]->(ss) return s,ss

match(s:Section),(ss:Staff) where s.name="Sofa" and ss.name="Satish" create(s)-[:handaledby]->(ss) return s,ss

create(c:Customer{name:"Pallawi"}) return c

create(c:Customer{name:"Niket"}) return c

create(c:Customer{name:"Shreya"}) return c

match(c:Customer),(f:Furniture) where c.name="Pallawi" and f.name="L-Shape Sofa" create (c)-[:enquire]->(f) return c,f

match(c:Customer),(f:Furniture) where c.name="Niket" and f.name="Steel cupboard" create (c)-[:enquire]->(f) return c,f

match(c:Customer),(f:Furniture) where c.name="Niket" and f.name="Coffee Table" create (c)-[:enquire]->(f) return c,f

match(c:Customer),(f:Furniture) where c.name="Shreya" and f.name="Coffee Table" create (c)-[:enquire]->(f) return c,f

match(c:Customer),(f:Furniture) where c.name="Shreya" and f.name="Coffee Table" create (c)-[:purchase]->(f) return c,f

match(c:Customer),(f:Furniture) where c.name="Pallawi" and f.name="L-Shape Sofa" create (c)-[:purchase]->(f) return c,f

#DipaliMeher

match(c:Customer),(f:Furniture) where c.name="Niket" and f.name="Coffee Table" create (c)-[:purchase]->(f) return c,f

Query 1)List the types of furnitures available in the showroom

match(s:Section),(f:Furniture) where (f)-[:belongsto]->(s)return f.name

Query2) List the sections handled by Mr. Satish

match(s:Section),(ss:Staff) where ss.name="Satish" and (s)-[:handaledby]->(ss) return s.name

Query 3)List the names of customers who have done only enquiry but not

made any purchase.

match(c:Customer),(f:Furniture) where (c)-[:enquire]->(f) and not(c)-[:purchase]->(f) return DISTINCT c.name

query4)List the fast-moving furniture types.

MATCH (c:Customer)-[:purchase]->(f:Furniture) with f.name as names, count(f.name) as count\_f WITH collect({names:names, count\_f:count\_f}) as rows, max(count\_f) as max UNWIND [row in rows WHERE row.count\_f = max] as row RETURN row.names as names, row.count\_f as count\_f

Model the following Hospitals information as a graph model, and answer the following queries using Cypher. Consider hospitals in and around Pune. Each hospital may have one or more specializations like Pediatric, Gynaec, Orthopedic, etc. A person can recommend/provide review for a hospital. A doctor can be associated with one or more hospitals.

match (d:Doctor),(h:Hospital) where h.name="Aundh CHS" and d.name="ssb" create (d)-[:visiting {day:["Mon","Tues"]}]->(h) return h,d

 1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model. [5]

2. Create nodes and relationships, along with their properties, and visualize your actual Graph model. [10]

 3. Answer the Queries

1. List the names of hospitals with pediatric specialization. [21/2]

* match(h:Hospital),(s:Spec) where s.name="Pediatric" and (h)-[:Specilizedin]->(s) return h.name

1. List the Names of doctors who are visiting “Jehangir Hospital” on Mondays. [21/2]

🡪 match (Doctor)-[v:visiting]->(Hospital{name:"Aundh CHS"})  where  "Mon" in v.day   return Doctor.name

1. List the most recommended Hospital for Gynaec specialization. [5]

🡪

#DipaliMeher

MATCH (h:Hospital)-[:Specilizedin]->(s:Spec) with s.name as names, count(s.name) as count\_vt WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

1. List the names of people who have given a rating of (>=3) for “Jehangir Hospital” [5]

🡪

match (person)-[v:review]->(Hospital{name:"Aundh CHS"}) where v.rate>=3 return person.name

[Note: You may Assume additional labels and properties depending on the query requirements]     
--

Database🡪

create(h:Hospital{name:"Jahangir",add:"Camp"}) return h

create(h:Hospital{name:"Sahyadri",add:"Deccan"}) return h

create(h:Hospital{name:"Aundh CHS",add:"Aundh"}) return h

create(s:Spec{name:"Pediatric"}) return s

create(s:Spec{name:"Gynac"}) return s

create(s:Spec{name:"Ortho"}) return s

match(h:Hospital),(s:Spec) where h.name="Sahyadri" and s.name="Gynac" create(h)-[:Specilizedin]->(s) return h,s

match(h:Hospital),(s:Spec) where h.name="Sahyadri" and s.name="Gynac" create(h)-[:Specilizedin]->(s) return h,s

match(h:Hospital),(s:Spec) where h.name="Aundh CHS" and s.name="Ortho" create(h)-[:Specilizedin]->(s) return h,s

match(h:Hospital),(s:Spec) where h.name="Aundh CHS" and s.name="Pediatric" create(h)-[:Specilizedin]->(s) return h,s

match(h:Hospital),(s:Spec) where h.name="Jahangir" and s.name="Pediatric" create(h)-[:Specilizedin]->(s) return h,s

create(p:person{name:"Dipali"}) return p

#DipaliMeher

create(p:person{name:"Ranjana"}) return p

create(p:person{name:"Niket"}) return p

match (p:person),(h:Hospital) where p.name="Ranjana" and h.name="Jahangir" create (p)-[:recommands]->(h) return p,h

match (p:person),(h:Hospital) where p.name="Niket" and h.name="Jahangir" create (p)-[:recommands]->(h) return p,h

match (p:person),(h:Hospital) where p.name="Niket" and h.name="Aundh CHS" create (p)-[:recommands]->(h) return p,h

match (p:person),(h:Hospital) where p.name="Niket" and h.name="Aundh CHS" create (p)-[:review {rate:4}]->(h) return p,h

match (p:person),(h:Hospital) where p.name="Dipali" and h.name="Aundh CHS" create (p)-[:review {rate:2}]->(h) return p,h

match (p:person),(h:Hospital) where p.name="Ranjana" and h.name="Jahangir" create (p)-[:review {rate:5}]->(h) return p,h

match (p:person),(h:Hospital) where p.name="Niket" and h.name="Jahangir" create (p)-[:review {rate:1}]->(h) return p,h

create( d:Doctor{name:"ssb"}) return d

create( d:Doctor{name:"Meenal"}) return d

create( d:Doctor{name:"Satish"}) return d

match(d:Doctor),(h:Hospital) where d.name="ssb" and h.name="Jahangir" create (d)-[ :associated]->(h) return d,h

match(d:Doctor),(h:Hospital) where d.name="Satish" and h.name="Jahangir" create (d)-[ :associated]->(h) return d,h

match(d:Doctor),(h:Hospital) where d.name="Meenal" and h.name="Aundh CHS" create (d)-[ :associated]->(h) return d,h

#DipaliMeher

match(d:Doctor),(h:Hospital) where d.name="Meenal" and h.name="Sahyadri" create (d)-[ :associated]->(h) return d,h

Query1) match(h:Hospital),(s:Spec) where s.name="Pediatric" and (h)-[:Specilizedin]->(s) return h.name

Query2) create visiting relationship

match (d:Doctor),(h:Hospital) where h.name="Aundh CHS" and d.name="ssb" create (d)-[:visiting {day:["Mon","Tues"]}]->(h) return h,d

match (d:Doctor),(h:Hospital) where h.name="Aundh CHS" and d.name="Satish" create (d)-[:visiting {day:["Wed","Fri"]}]->(h) return h,d

match (d:Doctor),(h:Hospital) where h.name="Aundh CHS" and (d)-[:visiting]->(h) return d.name

final query: match (Doctor)-[v:visiting]->(Hospital{name:"Aundh CHS"}) where "Mon" in v.day return Doctor.name

Query 3) Kindly check

match(p:person),(h:Hospital) where p.name="Niket"and h.name="Sahyadri" create (p)-[:recommands]->(h) return p,h

match(p:person),(h:Hospital) where p.name="Dipali"and h.name="Sahyadri" create (p)-[:recommands]->(h) return p,h

#DipaliMeher

FINAL QUERY

MATCH (p:person)-[:recommands]->(h:Hospital),(h:Hospital)-[:Specilizedin]->(s:Spec) where s.name="Gynac" with h.name as names, count(h.name) as count\_vt WITH collect({names:names, count\_vt:count\_vt}) as rows, max(count\_vt) as max UNWIND [row in rows WHERE row.count\_vt = max] as row RETURN row.names as names, row.count\_vt as count\_vt

Query 4) match (person)-[v:review]->(Hospital{name:"Jahangir"}) where v.rate>=3 return person.name