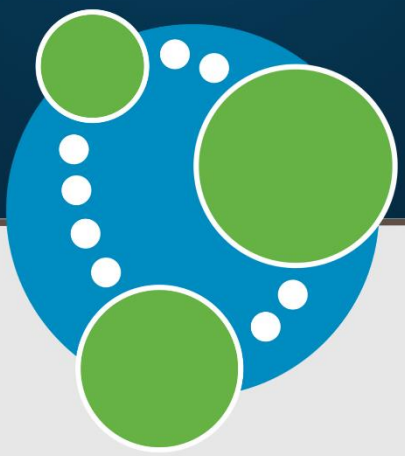
ISBN: 978-93-88901-93-2



Handbook of Case Studies in neo4j

Author:

Dr. Dipali P. Meher

Bhumi Publishing

Bhumi Publishing, India

Handbook of Case Studies in

(ISBN: 978-93-88901-93-2)

Author

Dr. Dipali P. Meher

(M.C.S., Ph.D.)

Assistant Professor and M.Sc. CS Course Coordinator,

Department of Computer Science,

Modern College of Arts, Science and Commerce,

Ganeshkhind, Pune 16

Bhumi Publishing

2023

First Edition: November, 2023

ISBN: 978-93-88901-93-2

Published by:

Bhumi Publishing,

Nigave Khalasa, Kolhapur 416207, Maharashtra, India

Website: www.bhumipublishing.com

E-mail: bhumipublishing@gmail.com

Book Available online at:

https://www.bhumipublishing.com/book/

PREFACE

At the end of 20th century everyone started using internet and web for their business growth and the word Bigdata came into picture. But relational databases unable to handle clustering concept of this word and NOSQL had taken birth. In 1995 Carlo Strozzi coined the term NOSQL and Now NOSQL databases boomed in all fields like Machine Learning, AI, Deep learning. Graph database is one of the NOSQL database which behaves like relational database and follows all properties of it. Though belonging in NOSQL database graph databases are called as odd man out fish in NOSQL pond because of their behavior like relational databases. Neo4j is a graph database management system developed by Neo4j Inc. in 2010. Graphs are a flexible data model that will meet all business needs. Year 2023 has been pivotal year in graph technology as their growth rate is 400 percent per year. In coming years Graph databases will replace relational databases.

This subject included in Post graduate studies of Computer Science and Applications. Study of this subject will be best to obtain Industrial Training or Industrial Project which is necessary for placement. This handbook will be useful for the students who want learn and solve complex problems in graph databases.

I extend my best wishes to the students who will embark on the journey of practical solutions in NOSQL databases. I hope that this book will serve as valuable resource for understanding the implementation of Graph databases using Neo4j.

- **Dr. Dipali P. Meher**

TABLE OF CONTENT

Chapter 1 Song-Author

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] →Recording Company

Recording Company →[Finances] →Song

You may add more labels and relationship and their properties, as per assumptions.

Node creation: Song Author

```
CREATE(:SongAuthor{Name:'Indiwar',followers:'50M'})
CREATE(:SongAuthor{Name:'Sahil',followers:'15M'})
CREATE(:SongAuthor{Name:'Gulzar',followers:'12M'})
CREATE(:SongAuthor{Name:'Shanta',followers:'14M'})
CREATE(:SongAuthor{Name:'Jagdish',followers:'4M'})
CREATE(:SongAuthor{Name:'Sameer', followers:'3M'})
```

Node creation: Song

```
CREATE(:Song{Name:'What is Love', likes:'40M'})
CREATE(:Song{Name:'Tuzse Naraj Nahi Jindgi', likes:'12M'})
CREATE(:Song{Name:'Mere Ghar Ayee EkNanhi Pari', likes:'7M'})
CREATE(:Song{Name:'Main Aur Mera Saya',likes:'8M'})
CREATE(:Song{Name:'Morya Morya', likes:'10M'})
CREATE(:Song{Name:'Asen Mi Nasen Mi', likes:'2.5M'})
CREATE(:Song{Name:'Mamachya Gawala Jauya', likes:'8M'})
CREATE(:Song{Name:'Ghal Ghal Pinga Wara', likes:'5M'})
```

Relationship: Written by

```
match(s:SongAuthor),(ss:Song)
where s.Name='Indiwar' and ss.Name='What is Love'
create(ss)-[:written_by]->(s)  return s,ss
```

```
match(s:SongAuthor),(ss:Song)
where s.Name='Sahil' and ss.Name='Tuzse Naraj Nahi Jindgi'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Gulzar' and ss.Name='Mere Ghar Ayee EkNanhi Pari'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Gulzar' and ss.Name='Main Aur Mera Saya'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Sahil' and ss.Name='Morya Morya'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Shanta' and ss.Name='Asen Mi Nasen Mi'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Sameer' and ss.Name='Mamachya Gawala Jauya'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Shanta' and ss.Name='Ghal Ghal Pinga Wara'
create(ss)-[:written_by]->(s)  return s,ss

match(s:SongAuthor),(ss:Song)
where s.Name='Jagdish' and ss.Name='What is Love'
create(ss)-[:written_by]->(s)  return s,ss
```

Node Creation: Artist

```
CREATE(:Artist{Name:'Pallawi'})
CREATE(:Artist{Name:'Kishor'})
CREATE(:Artist{Name:'Asha'})
CREATE(:Artist{Name:'Arun'})
```

Relationship creation: Performs

```
match(a:Artist),(ss:Song)
where a.Name='Kishor' and ss.Name='What is Love' create(a)-[:performs]->(ss)  return a,ss

match(a:Artist),(ss:Song)
where a.Name='Asha' and ss.Name='Tuzse Naraj Nahi Jindgi' create(a)-[:performs]->(ss)
return a,ss

match(a:Artist),(ss:Song)
where a.Name='Pallawi' and ss.Name='Mere Ghar Ayee EkNanhi Pari'
create(a)-[:performs]->(ss)  return a,ss

match(a:Artist),(ss:Song)
where a.Name='Kishor' and ss.Name='Main Aur Mera Saya'
create(a)-[:performs]->(ss)  return a,ss

match(a:Artist),(ss:Song)
where a.Name='Arun' and ss.Name='Morya Morya'
create(a)-[:performs]->(ss)  return a,ss

match(a:Artist),(ss:Song)
where a.Name='Pallawi' and ss.Name='Asen Mi Nasen Mi'
create(a)-[:performs]->(ss)  return a,ss

match(a:Artist),(ss:Song)
where a.Name='Pallawi' and ss.Name='Morya Morya'
create(a)-[:performs]->(ss)  return a,ss
```

```
match(a:Artist),(ss:Song)
where a.Name='Arun' and ss.Name='Mamachya Gawala Jauya'
create(a)-[:performs]->(ss)  return a,ss

match(a:Artist),(ss:Song)
where a.Name='Asha' and ss.Name='Ghal Ghal Pinga Wara'
create(a)-[:performs]->(ss)  return a,ss
```

Node Creation: Recording Studio

```
CREATE(:Recording_Studio{Name:'Sony Studio'})
CREATE(:Recording_Studio{Name:'Zee Studio'})
```

Relationship creation: recorded in

```
match(r:Recording_Studio),(ss:Song)
where r.Name='Sony Studio' and ss.Name='What is Love'
create(ss)-[:recordedin]->(r)  return ss,r

match(r:Recording_Studio),(ss:Song)
where r.Name='Sony Studio' and ss.Name='Tuzse Naraj Nahi Jindgi'
create(ss)-[:recordedin]->(r)  return ss,r

match(r:Recording_Studio),(ss:Song)
where r.Name='Sony Studio' and ss.Name='Mere Ghar Ayee EkNanhi Pari'
create(ss)-[:recordedin]->(r)  return ss,r

match(r:Recording_Studio),(ss:Song)
where r.Name='Sony Studio' and ss.Name='Main Aur Mera Saya'
create(ss)-[:recordedin]->(r)  return ss,r

match(r:Recording_Studio),(ss:Song)
where r.Name='Zee Studio' and ss.Name='Mamachya Gawala Jauya'
create(ss)-[:recordedin]->(r)  return ss,r
```

```
match(r:Recording_Studio),(ss:Song)
where r.Name='Zee Studio' and ss.Name='Morya Morya'
create(ss)-[:recordedin]->(r)  return ss,r

match(r:Recording_Studio),(ss:Song)
where r.Name='Zee Studio' and ss.Name='Asen Mi Nasen Mi'
create(ss)-[:recordedin]->(r)  return ss,r

match(r:Recording_Studio),(ss:Song)
where r.Name='Zee Studio' and ss.Name='Ghal Ghal Pinga Wara'
create(ss)-[:recordedin]->(r)  return ss,r
```

Node Creation: Recording Company

```
create(: Recording_Company{Name:'Sony ET'})
create(: Recording_Company{Name:'Zee ET'})
```

Relationship Creation: managed by

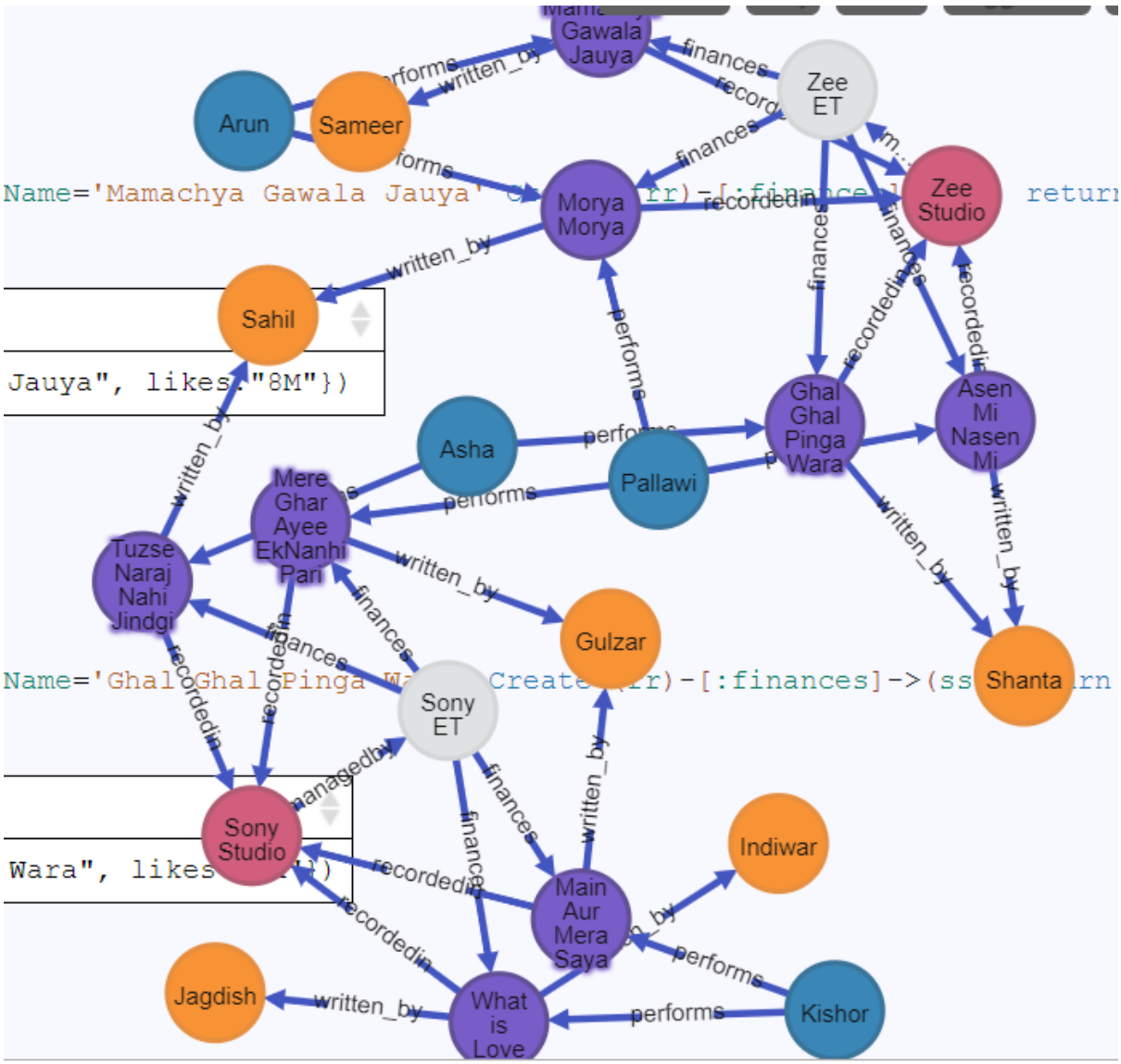
```
match(r:Recording_Studio),(rr:Recording_Company)
where r.Name='Zee Studio' and rr.Name='Zee ET'
create(r)-[:managedby]->(rr)  return rr,r

match(r:Recording_Studio),(rr:Recording_Company)
where r.Name='Sony Studio' and rr.Name='Sony ET'
create(r)-[:managedby]->(rr)  return rr,r
```

Relationship creation: finances

```
Match (rr:Recording_Company),(ss:Song)
Where rr.Name='Sony ET'  and ss.Name='What is Love'
Create (rr)-[:finances]->(ss) return rr,ss

Match (rr:Recording_Company),(ss:Song)
Where rr.Name='Sony ET'  and ss.Name='Tuzse Naraj Nahi Jindgi'
Create (rr)-[:finances]->(ss) return rr,ss
```

```
Match (rr:Recording_Company),(ss:Song)
Where rr.Name='Sony ET'  and ss.Name='Mere Ghar Ayee EkNanhi Pari'
Create (rr)-[:finances]->(ss) return rr,ss

Match (rr:Recording_Company),(ss:Song)
Where rr.Name= 'Sony ET' and ss.Name='Main Aur Mera Saya'
Create (rr)-[:finances]->(ss) return rr,ss

Match (rr:Recording_Company),(ss:Song)
Where rr.Name= 'Zee ET'  and ss.Name='Morya Morya'
Create (rr)-[:finances]->(ss) return rr,ss

Match (rr:Recording_Company),(ss:Song)
Where rr.Name= 'Zee ET'  and ss.Name='Asen Mi Nasen Mi'
Create (rr)-[:finances]->(ss) return rr,ss

Match (rr:Recording_Company),(ss:Song)
Where rr.Name='Zee ET'   and ss.Name='Mamachya Gawala Jauya'
Create (rr)-[:finances]->(ss) return rr,ss

Match (rr:Recording_Company),(ss:Song)
Where rr.Name='Zee ET'   and ss.Name='Ghal Ghal Pinga Wara'
Create (rr)-[:finances]->(ss) return rr,ss
```

Graph Model:



Queries:

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

Ans:

```
match(s:SongAuthor),(ss:Song)
where s.Name='Shanta' and (ss)-[:written_by]->(s)
return ss.Name
```

```
Query:
match(s:SongAuthor),(ss:Song)  where s.Name='Shanta' and (ss)-[:written_b
```

ss.Name
Asen Mi Nasen Mi
Ghal Ghal Pinga Wara

Query took 16 ms and returned 2 rows. Result Details

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

Ans: Match (rr:Recording_Company),(ss:Song)

Where ss.Name='Asen Mi Nasen Mi' and (rr)-[:finances]->(ss)

return rr.Name

```
Query:
Match (rr:Recording_Company),(ss:Song)  Where ss.Name='Asen Mi Nasen Mi' and
```

rr.Name
Zee ET

Query took 16 ms and returned 1 rows. Result Details

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

Ans:

match(a:Artist),(ss:Song)

where ss.Name='Morya Morya' and (a)-[:performs]->(ss)

return a.Name

```
Query:
match(a:Artist),(ss:Song)  where ss.Name='Morya Morya'  and (a)-[:performs]->
```

a.Name
Pallawi
Arun

Query took 20 ms and returned 2 rows. Result Details

4) Name the songs recorded by the studio "……."

Ans:

match(r:Recording_Studio),(ss:Song)

where r.Name='Zee Studio' and (ss)-[:recordedin]->(r)

return ss.Name

```
Query:
match(r:Recording_Studio),(ss:Song)  where r.Name='Zee Studio' and (ss)-[:
```

ss.Name
Morya Morya
Asen Mi Nasen Mi
Mamachya Gawala Jauya
Ghal Ghal Pinga Wara

Query took 16 ms and returned 4 rows. Result Details

5)List the names of artists who have sung only songs written by " "__

Ans:

match(a:Artist)-[r:performs]->(ss:Song),

(ss:Song)-[rr:written_by]->(s:SongAuthor{Name:"Gulzar"})

return a.Name

```
Query:
match(a:Artist)-[r:performs]->(ss:Song), (ss:Song)-[rr:written_by]->(s:Son
```

a.Name
Kishor
Pallawi

Query took 78 ms and returned 2 rows. Result Details

6) List the names of songs financed by Zee, and sung by__

Ans:

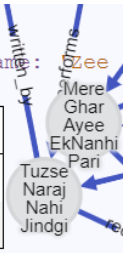
match(a:Artist {Name:'Pallawi' })-[:performs]->(ss:Song), (rr:Recording_Company{Name:

'Zee ET' })-[:finances]->(ss:Song) return a,ss

```
Query:
match(a:Artist {Name:'Pallawi' })-[:performs]->(ss:Song), (rr:Recording_Company{Name: 'Zee
```

a	ss
(54:Artist {Name:"Pallawi"})	(39:Song {Name:"Asen Mi Nasen Mi", likes:"2.5M"})
(54:Artist {Name:"Pallawi"})	(38:Song {Name:"Morya Morya", likes:"10M"})

Query took 1 ms and returned 2 rows. Result Details



Chapter 2 Employee Project

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

Node Creation: Employee

```
CREATE(:Employee {Name:'Dipali',  Qualification:['MCS','PhD'] })

CREATE(:Employee {Name: 'Satish', Qualification:['MCA','BCA','MSCIT']})

CREATE(:Employee {Name:'Meenal', Qualification:['B.Tech','MSCIT']})

CREATE(:Employee {Name:'Niket', Qualification:['B.Tech','M.Tech']})

CREATE(:Employee {Name:'Sheetal',Qualification:['M.Tech','B.Tech','MSCIT','MCS','BCS']})

CREATE(:Employee {Name:'Anand',  Qualification:['B.Tech','BCS','MSCIT']})

CREATE(:Employee {Name:'Shreya', Qualification:['B.Tech','M.Tech','MSCIT']})

CREATE(:Employee {Name:'Pallawi', Qualification:['PhD','BCS','MSC']})
```

CREATE(:Employee {Name:'Poonam', Qualification:['PhD','BCS','MSC']})

Node Creation: Department

CREATE(:Department{Name:'Computer Science'})

CREATE(:Department{Name:'Maths'})

CREATE(:Department{Name:'Electronics'})

CREATE(:Department{Name:'Statistics'})

Node Creation: Skillset

CREATE(:Skillset{skills:['Fluent Communication', 'Java Developer']})

CREATE(:Skillset{skills:['Java Developer', 'Leadership Qualities']})

CREATE(:Skillset{skills:['Leadership Qualities', 'Tester']})

CREATE(:Skillset{skills:['Optimistic', 'Python Developer']})

Node creation: Project

CREATE(:Project{Name:'College Website Design', Nod:10, client: 'ABC College'})

CREATE(:Project{Name:'Omega Development', Nod:65, client:' Raj Tutorials'})

CREATE(:Project{Name:'NpHard Problems', Nod:25, client:' PP Mind Solutions'})

CREATE(:Project{Name: 'Semiconductor Design', Nod:150, client: 'T Electronics'})

CREATE(:Project{Name:'Chatboat Development', Nod:13, client:' ASMA India'})

CREATE(:Project{Name:'Kappa Statistics', Nod:13, client:' ASMA India'})

Relationship: works in

Match(e:Employee),(d:Department)

where e.Name='Shreya' and

```
d.Name= 'Computer Science'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Poonam' and
d.Name= 'Computer Science'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Pallawi' and
d.Name= 'Computer Science'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Dipali' and
d.Name= 'Maths'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Satish' and
d.Name= 'Maths'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Meenal' and d.Name= 'Electronics'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Niket' and
d.Name= 'Electronics'
CREATE  (e)-[:Works_in]->(d)  return e,d
```

```
Match(e:Employee),(d:Department)
where e.Name='Shreya' and
d.Name= 'Statistics'
CREATE  (e)-[:Works_in]->(d)  return e,d

Match(e:Employee),(d:Department)
where e.Name='Anand' and d.Name= 'Computer Science'
CREATE  (e)-[:Works_in]->(d)  return e,d
```

Relationship Creation Has Acquired

```
Match(e:Employee),(s:Skillset)
where e.Name='Dipali' and
s.skills =['Fluent Communication', 'Java Developer']
CREATE (e)-[:Has_acquired]->(s) return e,s

Match(e:Employee),(s:Skillset)
where e.Name='Pallawi' and
s.skills =['Leadership Qualities' , 'Tester']
CREATE (e)-[:Has_acquired]->(s) return e,s

Match(e:Employee),(s:Skillset)
where e.Name='Satish' and
s.skills =['Optimistic', 'Python Developer']
CREATE (e)-[:Has_acquired]->(s) return e,s

Match(e:Employee),(s:Skillset)
where e.Name='Sheetal' and
s.skills =['Java Developer' , 'Leadership Qualities']
CREATE (e)-[:Has_acquired]->(s) return e,s
```

Relationship creation Assigned to

```
Match(e:Employee),(p:Project)
where e.Name='Sheetal' and
p.Name= 'College Website Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p
```

```
 Match(e:Employee),(p:Project)
where e.Name='Pallawi' and p.Name= 'College Website Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Dipali' and p.Name='Omega Development'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Satish' and p.Name='Omega Development'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Anand' and p.Name='College Website Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Shreya' and p.Name='College Website Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Meenal' and p.Name='Semiconductor Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Niket' and p.Name= 'Semiconductor Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p

Match(e:Employee),(p:Project)
where e.Name='Poonam' and
p.Name= 'College Website Design'
CREATE  (e)-[: Assigned_to]->(p)  return e,p
```

Relationship: Controlled BY

```
Match(p:Project),(d:Department)
where p.Name='College Website Design' and d.Name= 'Computer Science'
CREATE  (p)-[: Controlled_by] ->(d)  return p,d

Match(p:Project),(d:Department)
where p.Name='Semiconductor Design' and d.Name='Electronics'
CREATE  (p)-[: Controlled_by] ->(d)  return p,d

Match(p:Project),(d:Department)
where p.Name='NpHard Problems' and d.Name='Maths'
CREATE  (p)-[: Controlled_by] ->(d)  return p,d

Match(p:Project),(d:Department)
where p.Name='Chatboat Development' and d.Name='Computer Science'
CREATE  (p)-[: Controlled_by] ->(d)  return p,d

Match(p:Project),(d:Department)
where p.Name='Kappa Statistics' and d.Name='Statistics'
CREATE  (p)-[: Controlled_by] ->(d)  return p,d

Match(p:Project),(d:Department)
where p.Name='Omega Development'  and d.Name= 'Maths'
CREATE  (p)-[: Controlled_by] ->(d)  return p,d
```

Relationship Project Manager

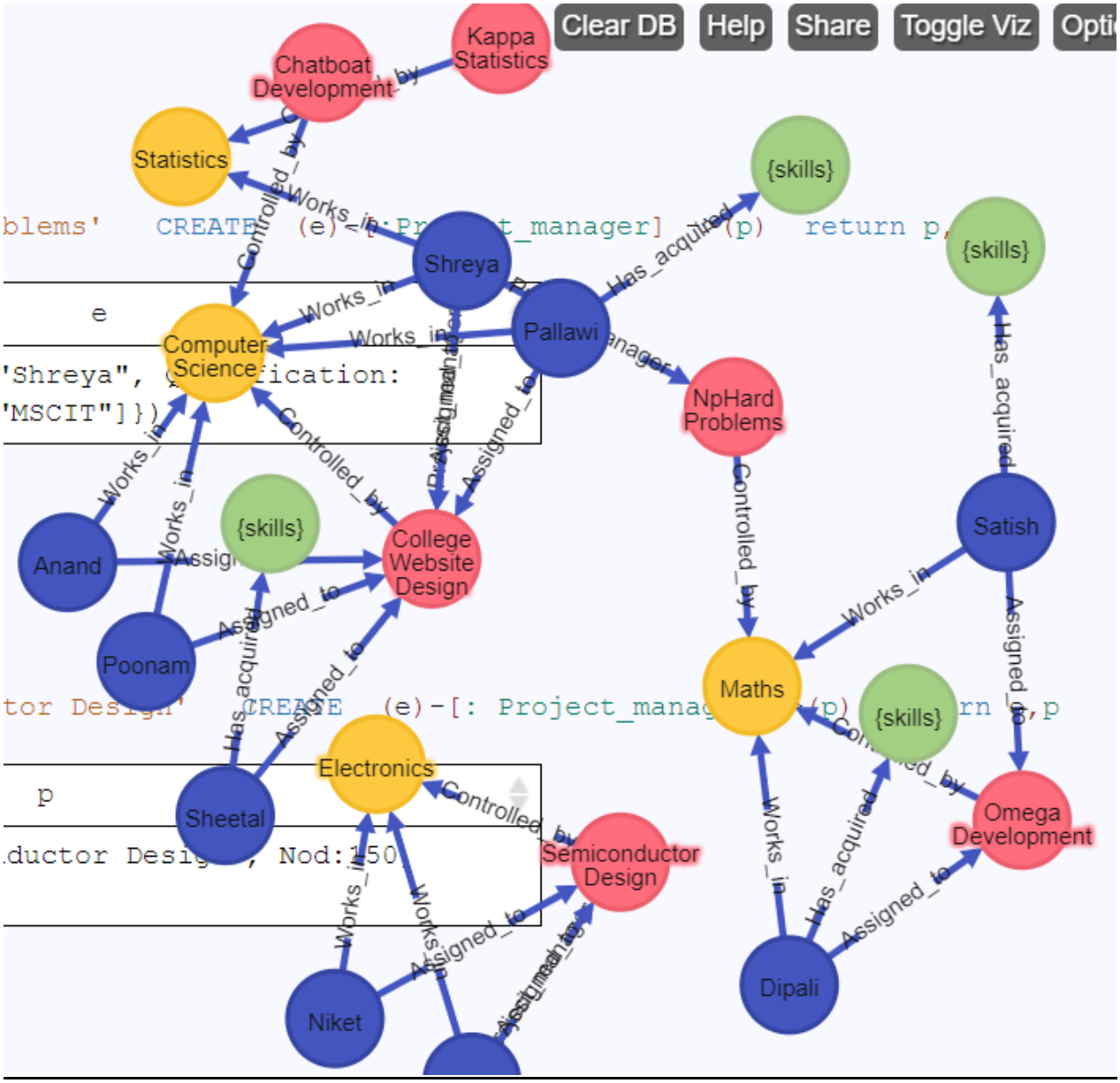
```
Match(e:Employee),(p:Project)
where e.Name='Shreya' and p.Name='College Website Design'
CREATE  (e)-[:Project_manager] ->(p)  return p,e

Match(e:Employee),(p:Project)
where e.Name='Shreya' and p.Name='NpHard Problems'
CREATE  (e)-[:Project_manager] ->(p)  return p,e
```

Match(e:Employee),(p:Project)

where e.Name='Meenal' and p.Name='Semiconductor Design'

CREATE (e)-[: Project_manager]->(p) return e,p

Graph:



Queries

a) List the Names of employees in department " "

Ans: for example computer science

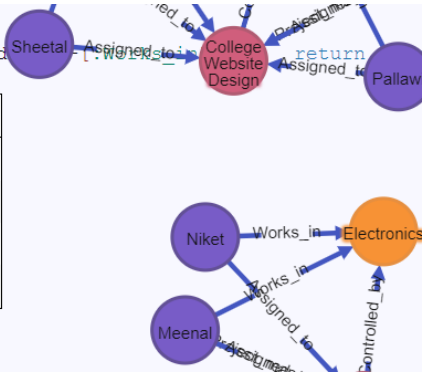
Match(e:Employee),(d:Department)

where d.Name= 'Computer Science' and (e)-[:Works_in]->(d) return e.Name

```
Query:
Match(e:Employee),(d:Department)  where d.Name= 'Computer Science'  and
```

e.Name
Anand
Shreya
Pallawi
Poonam

Query took 14 ms and returned 4 rows. Result Details



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

Ans:Match (p:Project),(d:Department)

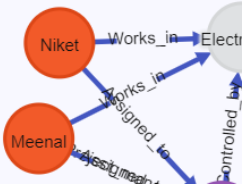
where d.Name='Electronics'

and(p)-[:Controlled_by]->(d) return p

```
Query:
Match (p:Project),(d:Department)  where d.Name='Electronics'  and(p)-[:Controlled_by]->(d) return p
```

p
(35:Project {Name:"Semiconductor Design", Nod:150, client:"T Electronics"})

Query took 13 ms and returned 1 rows. Result Details



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

Ans: Match(d:Department),(e:Employee)

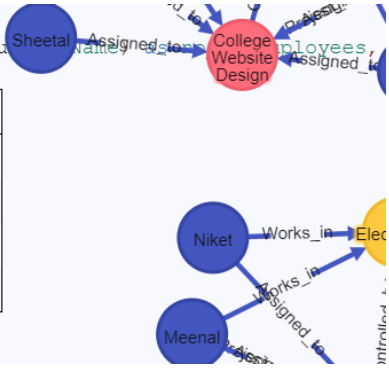
where (e)-[:Works_in]->(d)

return count(e.Name) as no_of_employees, d.Name

```
Query:
Match(d:Department),(e:Employee) where (e)-[:Works_in]->(d)  return cou
```

no_of_employees	d.Name
4	Computer Science
2	Maths
2	Electronics
1	Statistics

Query took 14 ms and returned 4 rows. Result Details



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

Ans: Match(e:Employee),(s:Skillset)

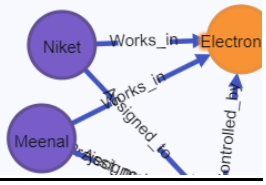
where e.Name='Dipali' and

(e)-[: Has_acquired]->(s) return s

```
Query:
Match(e:Employee),(s:Skillset) where e.Name='Dipali' and  (e)-[: Has_acquired]->(s)  return s
```

s
(62:Skillset {skills:["Fluent Communication","Java Developer"]})

Query took 12 ms and returned 1 rows. Result Details



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

Ans: Match(p:Project),(d:Department), (e:Employee)

where d.Name= 'Computer Science' and (p)-[: Controlled_by] ->(d)

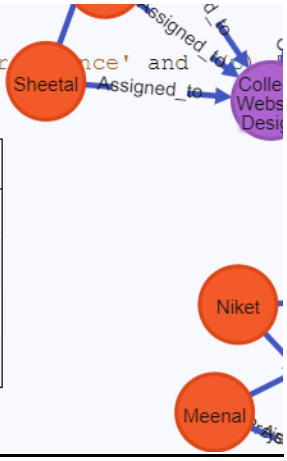
and (e)-[:Assigned_to]->(p) and (e)-[:Works_in]->(d)

return DISTINCT e.Name, p.Name

```
Query:
Match(p:Project),(d:Department), (e:Employee)  where  d.Name= 'Computer
>(p)  and (e)-[:Works_in]->(d)  return DISTINCT e.Name, p.Name
```

e.Name	p.Name
Anand	College Website Design
Shreya	College Website Design
Pallawi	College Website Design
Poonam	College Website Design

Query took 32 ms and returned 4 rows. Result Details



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

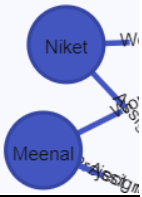
Ans:

MATCH (p:Project)-[:Controlled_by]->(d:Department), (e:Employee)-[:Project_manager]->(p:Project) where e.Name="Shreya" return p.Name

```
Query:
MATCH (p:Project)-[:Controlled_by]->(d:Department), (e:Employee)-[:Project_manager]->(p:Proje
```

p.Name
NpHard Problems
College Website Design

Query took 21 ms and returned 2 rows. Result Details



h)List the Names of the projects managed by employees as project managers

match (p:Project)-[:Project_manager]-(e:Employee) match (e)-[]-(d:Department) with p.Name as proj, e.Name as projMngr, d.Name as dept return DISTINCT proj,projMngr

```
Query:
match (p:Project)-[:Project_manager]-(e:Employee) match (e)-[]-(d:Depart
DISTINCT proj,projMngr
```

proj	projMngr
College Website Design	Shreya
NpHard Problems	Shreya
Semiconductor Design	Meenal

Query took 21 ms and returned 3 rows. Result Details

f) List All the projects and relationships with Departments and Employees.

Ans: match (p:Project)-[]-(d:Department) match (e:Employee)-[]-(d) return p.Name, d.Name, e.Name

```
Query:
match (p:Project)-[]-(d:Department) match (e:Employee)-[]-(d) return p.Name, d.
```

Show 10 entries Search:

p.Name	d.Name	e.Name
Chatboat Development	Computer Science	Anand
Chatboat Development	Computer Science	Pallawi
Chatboat Development	Computer Science	Poonam
Chatboat Development	Computer Science	Shreya
College Website Design	Computer Science	Anand
College Website Design	Computer Science	Pallawi
College Website Design	Computer Science	Poonam
College Website Design	Computer Science	Shreya
Omega Development	Maths	Satish
Omega Development	Maths	Dipali

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

Ans: match (e:Employee{name: 'Dipali Meher'})-[: Has_acquired]->(s:Skillset)
with s.Name as skill,
collect(distinct e.Name) as emp return skill, emp, size(emp) as empcnt
order by empcnt desc

Chapter 3 Furniture

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. Customer 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.
2. Create nodes and relationships, along with their properties, and visualize your actual Graph model.
3. Answer the queries.

a. List the types of furniture's available in the showroom.

b. List the sections handled by Mr. Satish.

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

d. List the fast-moving furniture types.

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

Node creation: Section

```
create(:Section{name:"Sofa"})
create(:Section{name:"Table"})
create(:Section{name:"Bed"})
create(:Section{name:"Cupboard"})
```

node creation: Furniture

```
create(:Furniture{name:"Coffee Table",color:"White",price:2000})
create(:Furniture{name:"Computer Table",color:"White",price:2000})
create(:Furniture{name:"L-Shape Sofa",color:"Brown",price:200000})
create(:Furniture{name:"Sofa cum bed",color:"green",price:240000})
create(:Furniture{name:"Steel cupboard",color:"pink",price:40000})
```

```
create(:Furniture{name:"bed with storage",color:"red",price:40000})
create(:Furniture{name:"bed without storage",color:"blue",price:40000})
```

relationship creation: belongs to

```
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

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
```

node creation: Staff

```
create(:Staff{name:"Satish"})
```

```
create(:Staff{name:"Meenal"})
create(:Staff{name:"Dipali"})
```

node creation: handled by

```
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

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

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

node creation: customer

```
create(:Customer{name:"Pallawi"})
create(:Customer{name:"Niket"})
create(:Customer{name:"Shreya"})
```

relationship creation: enquire

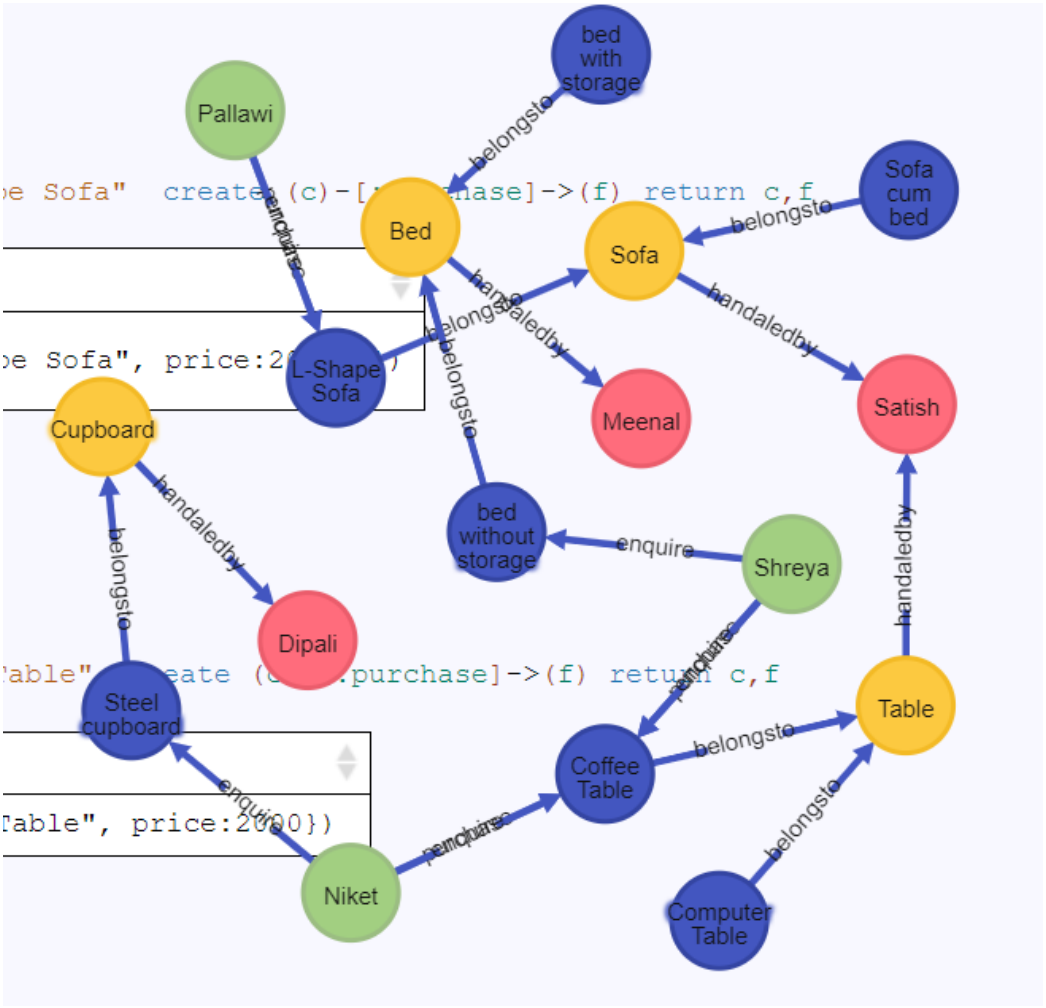
```
match(c:Customer),(f:Furniture)
where c.name="Pallawi" and f.name="L-Shape Sofa"
create (c)-[:enquire]->(f) return c,f

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="bed without storage"
create (c)-[:enquire]->(f) return c,f
```

relationship creation: purchase

```
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

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

Graph:



Query 1) List the types of furniture's available in the showroom

match(s:Section),(f:Furniture)

where (f)-[:belongsto]->(s) return f.name

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

f.name
L-Shape Sofa
Sofa cum bed
Coffee Table
Computer Table
bed with storage
bed without storage
Steel cupboard

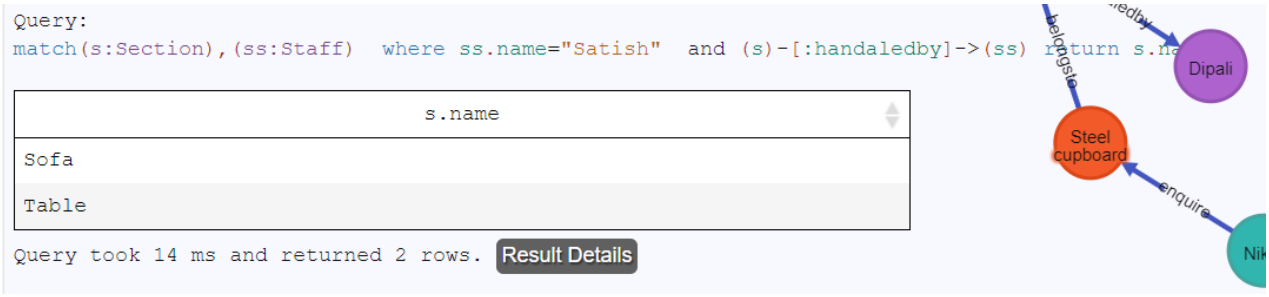
Query took 61 ms and returned 7 rows. Result Details

Query2) List the sections handled by Mr. Satish

match(s:Section),(ss:Staff)

where ss.name="Satish"

and (s)-[:handaledby]->(ss) return s.name

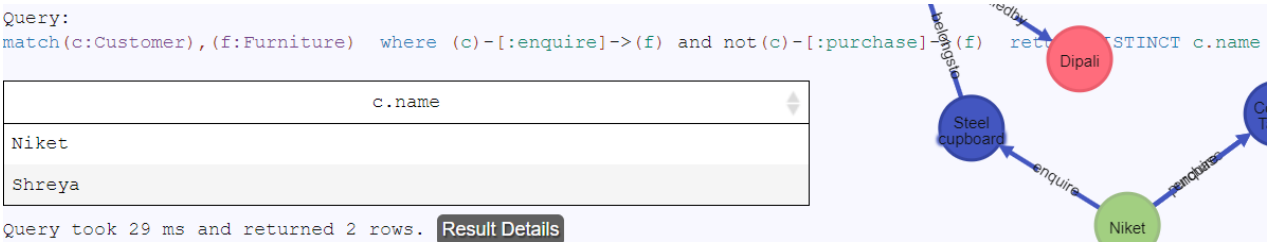


Query 3) List the names of customers who have done only enquiry but not made any purchase.

match(c:Customer),(f:Furniture)

where (c)-[:enquire]->(f) and not(c)-[:purchase]->(f)

return DISTINCT c.name



Query 4) List the fast-moving furniture types.

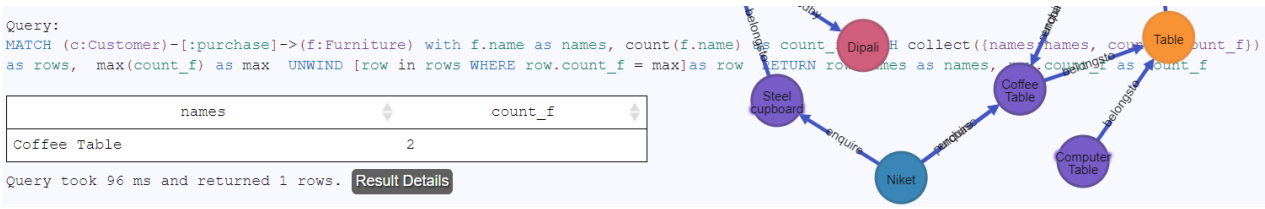
MATCH (c:Customer)-[:purchase]->(f:Furniture) with f.name as names, count(f.name) as count_f

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



Chapter 4 Doctor

Consider the doctors in and around pune. Each doctor is specialized in some stream like Pediatric, Gynaec, Heart, cancer, ENT, A doctor may be visiting doctor across many hospitals or he may owns a clinic. A person can provide a review/ can recommend a doctor.

Doctor Node

create(:Doctor{name:"Dr. Dipali Meher"})
create(:Doctor{name:"Dr. Uma Kulkarni"})
create(:Doctor{name:"Dr. Pallawi Bulakh"})

Stream Node

create(:Stream{name:"Pediatric"})
create(:Stream{name:"Gynaec"})
create(:Stream{name:"Heart Spec"})
create(:Stream{name:"Cancer Spec"})
create(:Stream{name:"ENT"})

Relationship: specialized in

match(d:Doctor),(s:Stream) where d.name="Dr. Dipali Meher" and s.name="Pediatric" create (d)-[:Specilized_in]->(s) return d,s

match(d:Doctor),(s:Stream) where d.name="Dr. Pallawi Bulakh" and s.name="ENT" create (d)-[:Specilized_in]->(s) return d,s

match(d:Doctor),(s:Stream) where d.name="Dr. Uma Kulkarni" and s.name="Gynaec" create (d)-[:Specilized_in]->(s) return d,s

match(d:Doctor),(s:Stream) where d.name="Dr. Pallawi Bulakh" and s.name="Gynaec" create (d)-[:Specilized_in]->(s) return d,s

node creation: Hospital

create(:Hospital{name:"Sahyadri",add:"Kothrud"})

```
create(:Hospital{name:"Jahangir",add:"Camp"})
create(:Hospital{name:"Sasoon",add:"Pune Station"})
```

relationship: visiting

```
match(d:Doctor),(h:Hospital) where d.name="Dr. Uma Kulkarni" and h.name="Sahyadri"
create  (d)-[:visiting {day:['Mon','Tues']}]->(h) return d,h

match(d:Doctor),(h:Hospital) where d.name="Dr. Pallawi Bulakh" and h.name="Sahyadri"
create  (d)-[:visiting {day:['Wed','Tues']}]->(h) return d,h

match(d:Doctor),(h:Hospital) where d.name="Dr. Pallawi Bulakh" and h.name="Jahangir"
create  (d)-[:visiting {day:['Fri','Sat']}]->(h) return d,h
```

node creation: clinic

```
create(:Clinic{name:"Anvi Clinic",add: "Kothrud"})
create(:Clinic{name:"Prasad Clinic",add: "Pashan"})
create(:Clinic{name:"Jay Clinic",add: "Camp"})
```

relationship: owns

```
match(d:Doctor),(c:Clinic) where d.name="Dr. Pallawi Bulakh" and c.name="Prasad Clinic"
create (d)-[:owns]->(c) return d,c

match(d:Doctor),(c:Clinic) where d.name="Dr. Dipali Meher" and c.name="Anvi Clinic"
create (d)-[:owns]->(c) return d,c

match(d:Doctor),(c:Clinic) where d.name="Dr. Dipali Meher" and c.name="Jay Clinic"
create (d)-[:owns]->(c) return d,c
```

node creation: person

```
create(:Person{name:"Meenal"})
create(:Person{name:"Prakash"})
create(:Person{name:"Satish"})
create(:Person{name:"Niket"})
```

node creation: recommends

match(p:Person),(d:Doctor) where d.name="Dr. Dipali Meher" and p.name="Meenal" create (p)-[:recommends]->(d) return d,p

match(p:Person),(d:Doctor) where d.name="Dr. Pallawi Bulakh" and p.name="Meenal" create (p)-[:recommends]->(d) return d,p

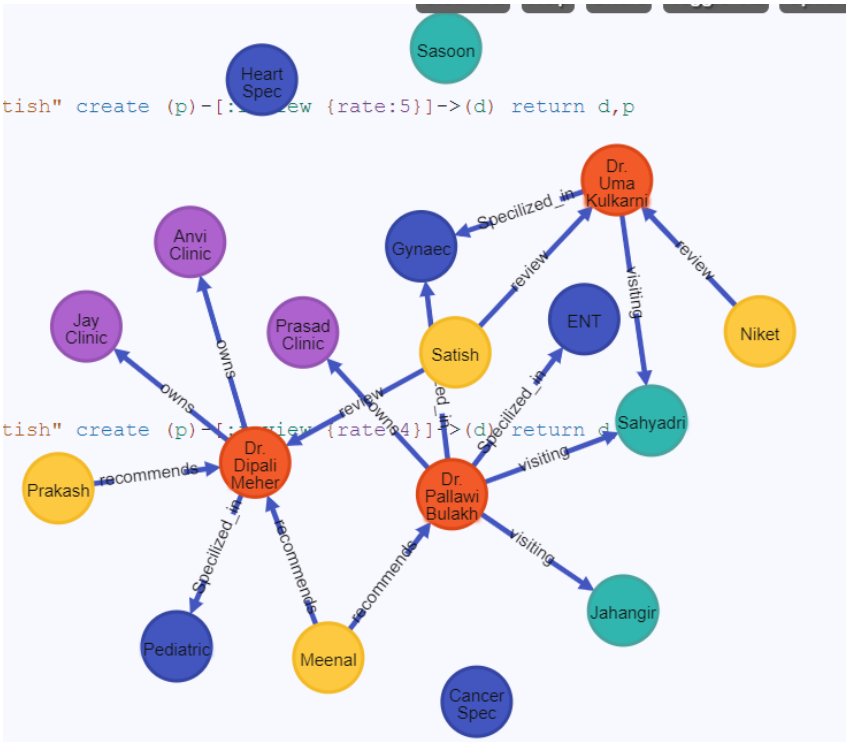
match(p:Person),(d:Doctor) where d.name="Dr. Dipali Meher" and p.name="Prakash" create (p)-[:recommends]->(d) return d,p

match(p:Person),(d:Doctor) where d.name="Dr. Dipali Meher" and p.name="Satish" create (p)-[:review {rate:5}]->(d) return d,p

match(p:Person),(d:Doctor) where d.name="Dr. Uma Kulkarni" and p.name="Satish" create (p)-[:review {rate:4}]->(d) return d,p

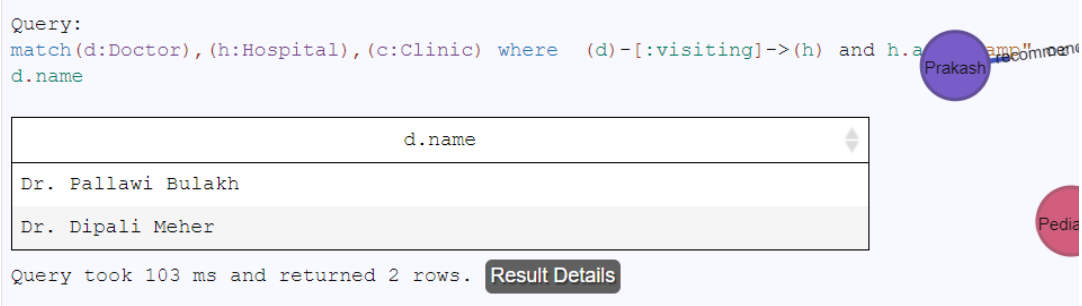
match(p:Person),(d:Doctor) where d.name="Dr. Uma Kulkarni" and p.name="Niket" create (p)-[:review {rate:3}]->(d) return d,p

Graph:



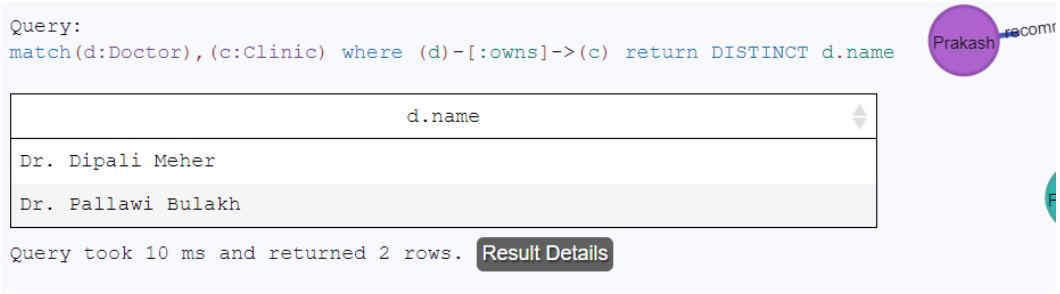
1. **List the Paediatric doctors in camp area**

   Ans: match(d:Doctor),(h:Hospital),(c:Clinic) where (d)-[:visiting]->(h) and h.add="Camp" or (d)-[:owns]->(c) and c.add="Camp" return DISTINCT d.name



2. **List the doctors who owns a clinic**

   Ans: :match(d:Doctor),(c:Clinic) where (d)-[:owns]->(c) return DISTINCT d.name



3. **List the most recommended Gynaec in kothrud**

   Ans: match (p:Person)-[:recommends]->(d:Doctor), (d:Doctor)-[:visiting]->(h:Hospital) ,
   (d:Doctor)-[:owns]->(c:Clinic),
   (d:Doctor)-[:Specilized_in]->(s:Stream) where c.add="Kothrud" or h.add="Kothrud" and s.name="Gynaec" with d.name as names, count(d.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:
match (p:Person)-[:recommends]->(d:Doctor), (d:Doctor)-[:visiting]->(h:Hos
[:Specilized_in]->(s:Stream)  where c.add="Kothrud" or h.add="Kothrud" and
WITH collect({names:names, count_vt:count_vt}) as rows, max(count_vt) as r
row.names as names, row.count_vt as count_vt
```

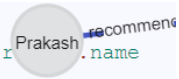
names	count_vt
Dr. Pallawi Bulakh	1

Query took 71 ms and returned 1 rows. Result Details

4. List all the reviews for " Dr. Kulkarni"

Ans: match (p:Person)-[v:review]->(d:Doctor{name:"Dr. Uma Kulkarni"}) return v.rate,p.name

```
Query:
match (p:Person)-[v:review]->(d:Doctor{name:"Dr. Uma Kulkarni"}) return v.r
```



v.rate	p.name
3	Niket
4	Satish

Query took 9 ms and returned 2 rows. Result Details

Chapter 5 Dairy

Model the following Dairy Brand information as a graph model, and answer the following queries using Cypher. There are various dairy brands like Amul, Go, Britannia, Gokul etc. Their popularity varies across different states in India. The popularity is measured as %, with a high popularity defined as >=90%, Medium Popularity between 50 to 90%, and Low popularity <50%. Each brand manufactures various types of Dairy products like milk, butter, cheese, curd etc. The milk product can be categorized into low fat/medium fat or high fat content type.

1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model.
2. Create nodes and relationships, along with their properties, and visualize your actual Graph model.
3. Answer the following queries using Cypher:
   a. List the names of different brands considered in your graph.
   b. List the brands that are highly popular in Maharashtra.
   c. List the popular cheese brands in Gujarat.
   d. List the brands manufacturing “low” fat content milk.

Node Creation: Brand

```
create(:Brand{name:"Amul"})
create(:Brand{name:"Britannia"})
create(:Brand{name:"Gokul"})
```

Node Creation: Product

```
create (:Product{name:"Milk"})
create (:Product{name:"Butter"})
create (:Product{name:"Cheese"})
create (:Product{name:"Curd"})
```

Node Creation: State

```
create (:State{name:"Maharashtra"})
create (:State{name:"Gujarat"})
create (:State{name:"Rajasthan"})
```

Relationship creation: Produces

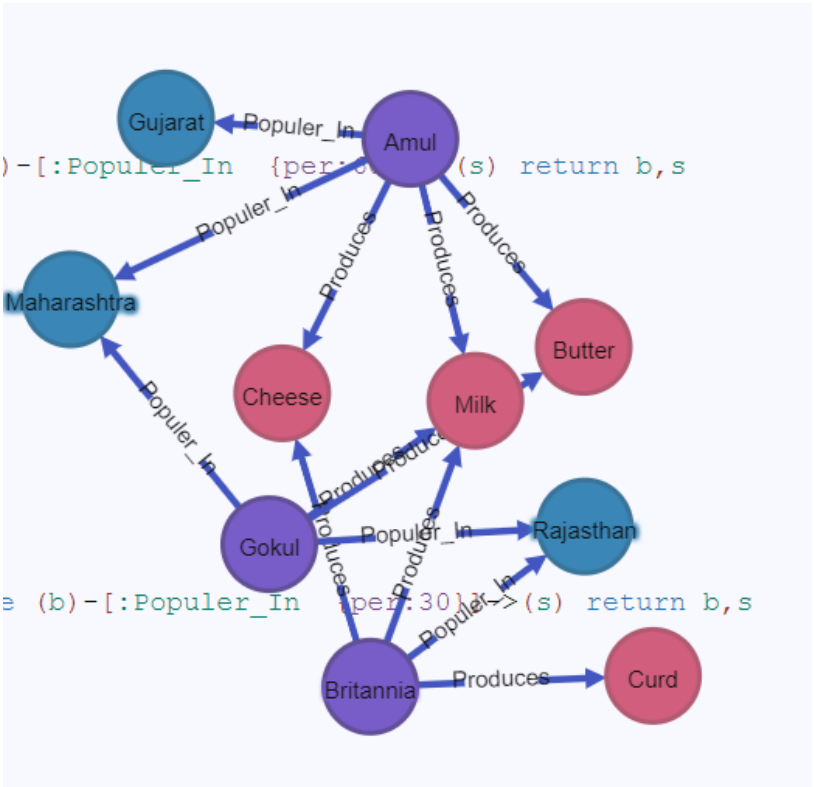
```
match(b:Brand),(p:Product)
where b.name="Amul" and p.name="Butter"
create (b)-[:Produces]->(p)  return b,p

match(b:Brand),(p:Product)
where b.name="Amul" and p.name="Cheese"
create (b)-[:Produces]->(p)  return b,p

match(b:Brand),(p:Product)
where b.name="Amul" and p.name="Milk"
create (b)-[:Produces{cat:[ "High Fat","Medium Fat"]}]->(p)  return b,p

match(b:Brand),(p:Product)
where b.name="Britannia" and p.name="Curd"
create (b)-[:Produces]->(p)  return b,p

match(b:Brand),(p:Product)
 where b.name="Britannia" and p.name="Milk"
create (b)-[:Produces {cat:[ "Low Fat","High Fat"]}]->(p)  return b,p

match(b:Brand),(p:Product)
 where b.name="Britannia" and p.name="Cheese"
create (b)-[:Produces ]->(p)  return b,p

match(b:Brand),(p:Product)
where b.name="Gokul" and p.name="Milk"
create (b)-[:Produces {cat: ["Low Fat"]}]->(p)  return b,p
match(b:Brand),(p:Product)
where b.name="Gokul" and p.name="Butter"
create (b)-[:Produces]->(p)  return b,p
```

Relationship: popular in

```
match(s:State),(b:Brand)
where b.name="Amul" and s.name="Maharashtra"
create (b)-[:Populer_In  {per:50}]->(s) return b,s
```

```
match(s:State),(b:Brand)
where b.name="Amul" and s.name="Gujarat"
create (b)-[:Populer_In  {per:98}]->(s) return b,s

match(s:State),(b:Brand)
where b.name="Gokul" and s.name="Maharashtra"
create (b)-[:Populer_In  {per:95}]->(s) return b,s

match(s:State),(b:Brand)
where b.name="Gokul" and s.name="Rajasthan"
create (b)-[:Populer_In  {per:60}]->(s) return b,s

match(s:State),(b:Brand)
where b.name="Britannia" and s.name="Rajasthan"
create (b)-[:Populer_In  {per:30}]->(s) return b,s
```

Graph:



Queries:

<u>1)List of different Brands</u>

```
match(b:Brand) return b.name
```

```
Query:
match(b:Brand) return b.name
```

b.name
Amul
Britannia
Gokul

Query took 8 ms and returned 3 rows. Result Details

2)List the brand that are highly popular in maharashtra

match(State{name:"Maharashtra"})<-[a:Populer_In]-(Brand)
where a.per> 90 return Brand.name

```
Query:
match(State{name:"Maharashtra"})<-[a:Populer_In]-(Brand)  where a.per> 90 return Brand.name
```

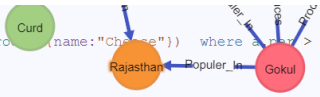
Brand.name
Gokul

Query took 109 ms and returned 1 rows. Result Details

3) highly popular cheese brands in Gujarat

match(State{name:"Gujarat"})<-[a:Populer_In]-(b:Brand)
,(b:Brand)-[:Produces]->(Product{name:"Cheese"})
where a.per > 90 return DISTINCT b.name

```
Query:
match(State{name:"Gujarat"})<-[a:Populer_In]-(b:Brand) ,(b:Brand)-[:Produces]->(Pro
b.name
```



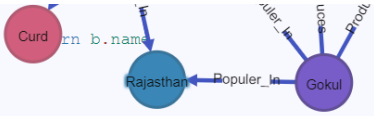
b.name
Amul

Query took 66 ms and returned 1 rows. Result Details

4) List the brands manufacturing “low” fat content milk

match(b:Brand) -[pr:Produces]->(p:Product{name: "Milk"})
where "Low Fat" in pr.cat return b.name

```
Query:
match(b:Brand) -[pr:Produces]->(p:Product{name: "Milk"}) where "Low Fat" in pr.cat
```



b.name
Gokul
Britannia

Query took 19 ms and returned 2 rows. Result Details

Chapter 6 Hospital

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.

1. Identify the labels and relationships, along with their properties, and draw a high-level Graph model.
2. Create nodes and relationships, along with their properties, and visualize your actual Graph model.

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

Node Creation: Hospital

```
create(:Hospital{name:"Jahangir",add:"Camp"})
create(:Hospital{name:"Sahyadri",add:"Deccan"})
create(:Hospital{name:"Aundh CHS",add:"Aundh"})
```

Node Creation: Specilization

```
create(:Spec{name:"Pediatric"})
create(:Spec{name:"Gynac"})
create(:Spec{name:"Ortho"})
```

relationship cration: specializedin

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

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

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

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

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

Node Creation: Person

```
create(:person{name:"Dipali"})
create(:person{name:"Ranjana"})
create(:person{name:"Niket"})
```

Relationship Creation: recommends

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

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

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

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

Relationship Creation: review and rating

```
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

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

Node Creation: Doctor

```
create(:Doctor{name:"Shubhangi"})
create(:Doctor{name:"Meenal"})
create(:Doctor{name:"Satish"})
```

Relationship Creation: associated

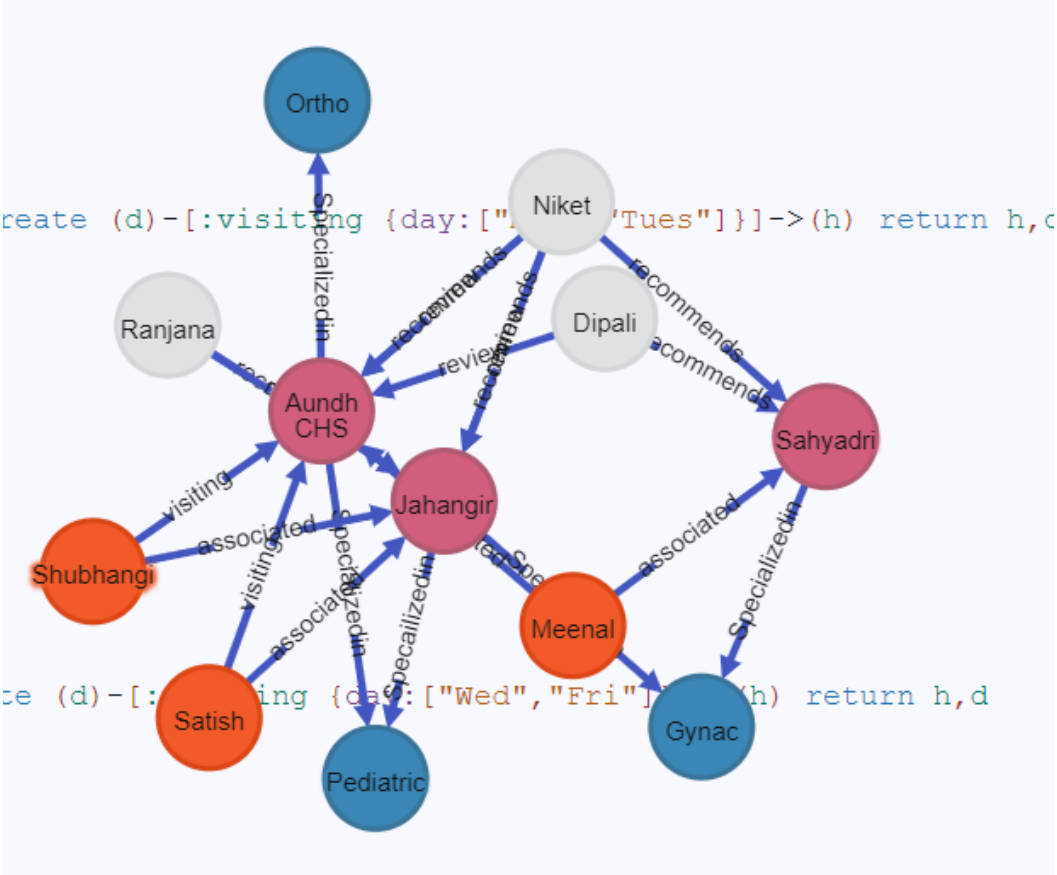
```
match(d:Doctor),(h:Hospital)
where d.name="Shubhangi" and h.name="Jahangir"
create (d)-[ :associated]->(h) return d,h

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

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

Relationship Creation: visited

```
match (d:Doctor),(h:Hospital)
where h.name="Aundh CHS" and d.name="Shubhangi"
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
```

Graph:



Queries:

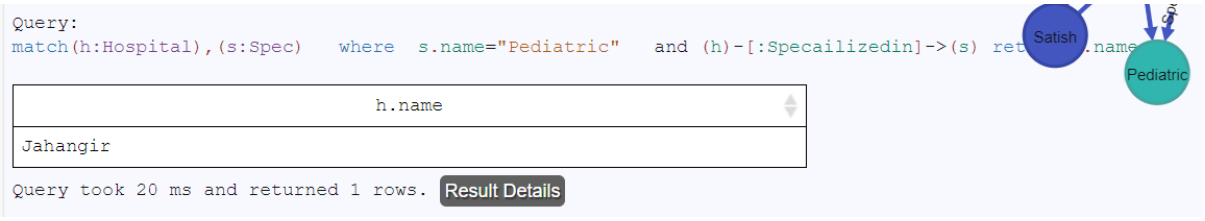
Query1) List the names of hospitals with pediatric specialization

Ans→

match(h:Hospital),(s:Spec)

where s.name="Pediatric"

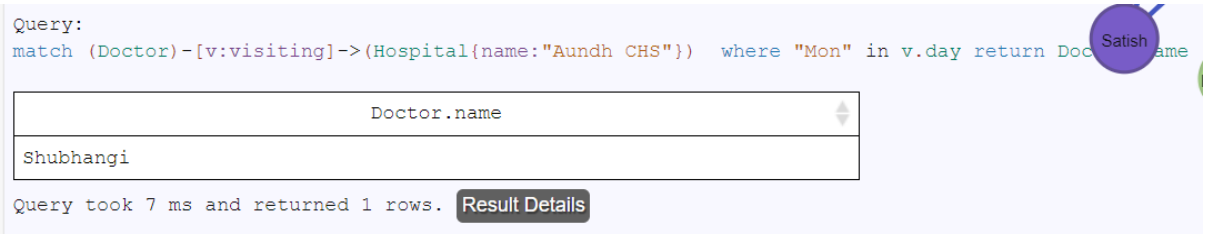
and (h)-[:Specailizedin]->(s) return h.name



Query2) List the Names of doctors who are visiting "Aundh Hospital" on Mondays.

Ans→ match (Doctor)-[v:visiting]->(Hospital{name:"Aundh CHS"})

where "Mon" in v.day return Doctor.name



Query 3) List the most recommended Hospital for Gynaec specialization.

MATCH (p:person)-[:recommends]->(h:Hospital),

(h:Hospital)-[:Specailizedin]->(s:Spec)

where s.name="Gynac" with h.name as names,

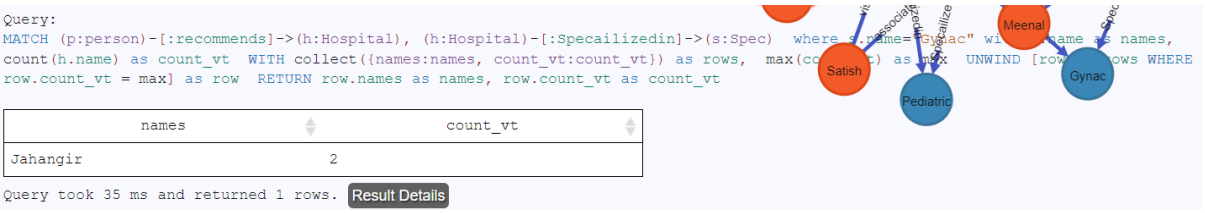
count(h.name) as count_vt

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) List the names of people who have given a rating of (>=3) for "Jehangir Hospital

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

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

Satish

person.name
Ranjana

Query took 17 ms and returned 1 rows. Result Details

Chapter 7 Hotel

Model the following Hotel information as a graph model and answer the following queries using cypher

Consider hotels in pune. Some hotels provide lodging facilities whereas some provide only restaurant facilities and some provide both. A person can rate(1-5 stars) a hotel for its facility/facilities. A person can recommend a hotel to his/her friends. A person can provide a review for hotel after his stay visit

Node Creation: Hotel

```
create (:Hotel{name:"New Poona Cafe",address:"Camp"})
create (:Hotel{name:"Modern Cafe",address:"Camp"})
create (:Hotel{name:"Prime Hotel",address:"Koregaon Park"})
create (:Hotel{name:"Poonam Hotel",address:"Deccan"})
```

Node Creation:Facility

```
create(:Facility{name:"Lodging"})
create(:Facility{name:"Restaurant"})
```

Relationship Creation: Provides

```
match(h:Hotel),(f:Facility)
where h.name="New Poona Cafe" and f.name="Lodging"
create (h)-[:provides]->(f) return h,f

match(h:Hotel),(f:Facility)
where h.name="New Poona Cafe" and f.name="Restaurant"
create (h)-[:provides]->(f) return h,f

match(h:Hotel),(f:Facility)
where h.name="Modern Cafe" and f.name="Restaurant"
create (h)-[:provides]->(f) return h,f

match(h:Hotel),(f:Facility)
where h.name="Poonam Hotel" and f.name="Restaurant"
create (h)-[:provides]->(f) return h,f
```

```
match(h:Hotel),(f:Facility)
where h.name="Prime Hotel" and f.name="Lodging"
create (h)-[:provides]->(f) return h,f
```

Node Creation: Person

```
create(:Person{name:"Dipali"})
create(:Person{name:"Prakash"})
create(:Person{name:"Shreya"})
create(:Person{name:"Meenal"})
create(:Person{name:"Pallawi"})
```

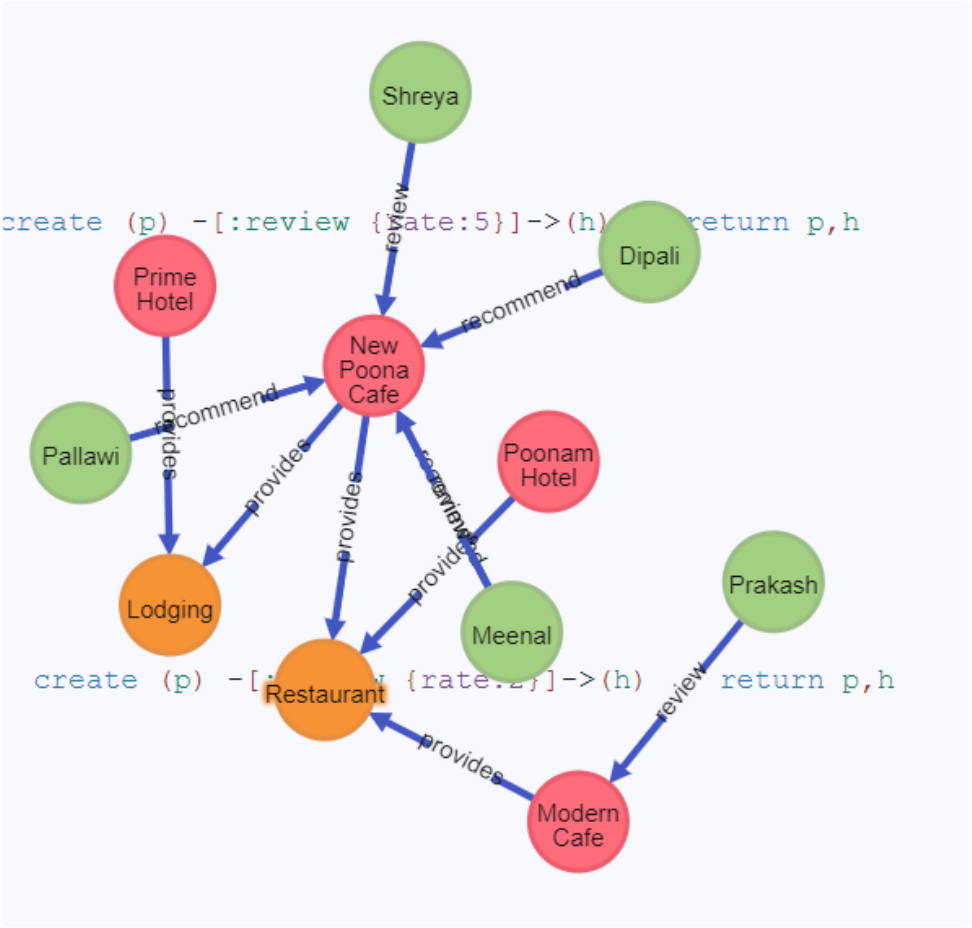
Relationship Creation: recommend

```
match (p:Person) , (h:Hotel)
where p.name="Dipali" and h.name="New Poona Cafe"
create (p) -[:recommend]->(h)    return p,h

match (p:Person) , (h:Hotel)
where p.name="Pallawi" and h.name="New Poona Cafe"
create (p) -[:recommend]->(h)    return p,h

match (p:Person) , (h:Hotel)
where p.name="Meenal" and h.name="New Poona Cafe"
create (p) -[:recommend]->(h)    return p,h
```

Relationship Creation: review/rate

```
match (p:Person) , (h:Hotel)
where p.name="Shreya" and h.name="New Poona Cafe"
create (p) -[:review {rate:4}]->(h)    return p,h

match (p:Person) , (h:Hotel)
where p.name="Prakash" and h.name="Modern Cafe"
create (p) -[:review {rate:5}]->(h)    return p,h

match (p:Person) , (h:Hotel)
where p.name="Meenal" and h.name="New Poona Cafe"
create (p) -[:review {rate:2}]->(h)    return p,h
```

Neo4j Model



Queries:

Query1) List the names of hotel in camp area

Match (h:Hotel) where h.address="Camp" return h.name

```
Query:
Match (h:Hotel) where h.address="Camp"  return h.name
```

h.name
New Poona Cafe
Modern Cafe

Query took 10 ms and returned 2 rows. Result Details

Query 2)List the names of hotels having both lodging and restaurant facility

MATCH

(:Facility{name:"Lodging"})--(n:Hotel)--(:Facility{name:"Restaurant"})

RETURN n.name

```
Query:
MATCH  (:Facility{name:"Lodging"})--(n:Hotel)--(:Facility{name:"Restaurant"})  RETURN n.name
```

n.name
New Poona Cafe

Query took 859 ms and returned 1 rows. Result Details

Query 3) List the names of hotels with high rating>04

match (Person)-[v:review]->(Hotel)

where v.rate>4 return DISTINCT Hotel.name

```
Query:
match (Person)-[v:review]->(Hotel)  where v.rate>4 return DISTINCT Hotel.name
```

Hotel.name
Modern Cafe

Query took 10 ms and returned 1 rows. Result Details

Query 4) List most recommended hotels in camp area

MATCH (p:Person)-[:recommend]->(h:Hotel)

where h.address="Camp" with h.name as names,

count(h.name) as count_vt

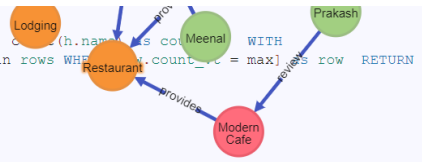
WITH collect({names:names, count_vt:count_vt}) as rows,

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:
MATCH (p:Person)-[:recommend]->(h:Hotel)  where h.address="Camp" 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
```



names	count_vt
New Poona Cafe	3

Query took 137 ms and returned 1 rows. Result Details

Chapter 8 Person

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.
2. Create nodes and relationships, along with their properties, and visualize your actual Graph model
3. Answer the following Queries in Cypher:

a. List the names of people who are parents.

b. List the names of people working on 'Finance 'project

c. List the names of people staying in 'Pune' and 'Mumbai'.

d. List the names of people who are mothers.

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

Node Creation: Person

```
create(:Person{name:"Dipali",age:39})
create(:Person{name:"Pallawi",age:43})
create(:Person{name:"Meenal",age:38})
create(:Person{name:"Prakash",age:40})
create(:Person{name:"Mrunu",age:2})
create(:Person{name:"Rugved",age:4})
create(:Person{name:"Shreya",age:2})
create(:Person{name:"Omkar",age:4})
create(:Person{name:"Kaustubh",age:45})
```

Relationship Creation: Friend_of

```
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
```

Relationship Creation: siblings (sisterof)

```
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="Shreya" and pp.name="Omkar"
create(p)-[:sisterof]->(pp)return p,pp
```

Relationship Creation: siblings (brotherof)

```
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="Rugved" and pp.name="Mrunu"
create(p)-[:brotherof]->(pp)return p,pp
```

Relationship Creation: Parents (motherof)

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

Relationship Creation: Parents (fatherof)

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

Node Creation: City

```
create(:City{name:"Pune"})
create(:City{name:"Mumbai"})
create(:City{name:"Kolhapur"})
```

Relationship Creation: staysin

```
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
```

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

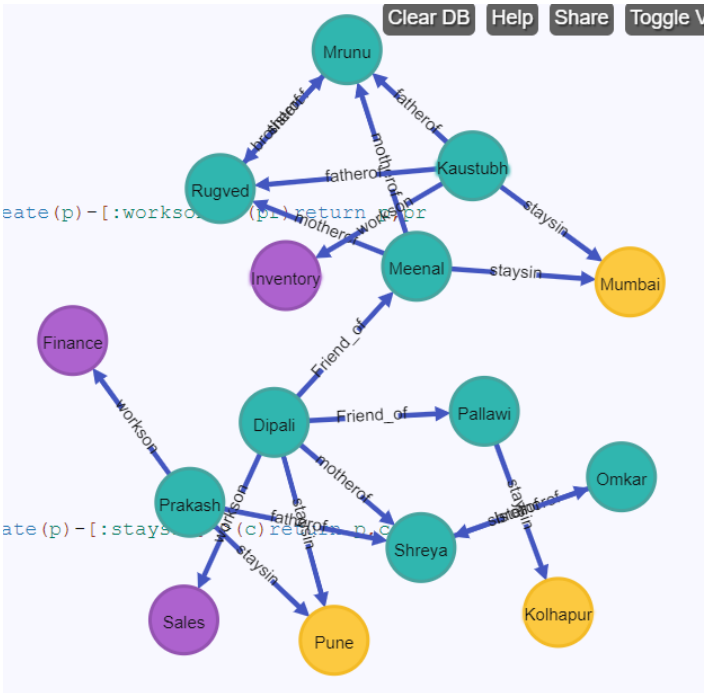
Node Creation: Project

```
create(:Project{name:"Finance"})
create(:Project{name:"Inventory"})
create(:Project{name:"Sales"})
```

Relationship Creation: workson

```
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
```

Neo4j Model:



Queries

1) List the names of people who are parents

→MATCH (p:Person),(pp:Person)

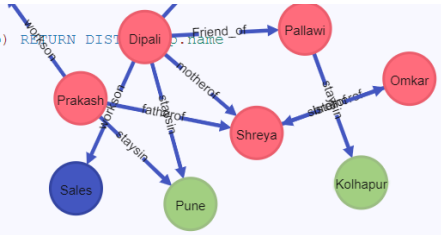
Where (p)-[:fatherof]->(pp) or (p)-[:motherof]->(pp)

RETURN DISTINCT p.name

```
Query:
MATCH (p:Person),(pp:Person)  Where  (p)-[:fatherof]->(pp) or (p)-[:motherof]->(pp) RETURN DISTINCT p.name
```

p.name
Dipali
Meenal
Prakash
Kaustubh

Query took 903 ms and returned 4 rows. Result Details



2) List the names of people working on 'Finance 'project

→ match(p:Person),(pr:Project)

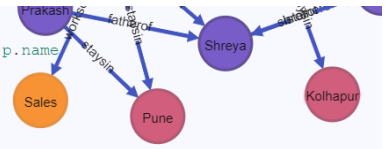
where pr.name="Finance" and (p)-[:workson]->(pr)

return p.name

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

p.name
Prakash

Query took 20 ms and returned 1 rows. Result Details



3) List the names of people staying in 'Pune' and 'Mumbai'.

→ MATCH (p:Person),(c:City)

WHERE c.name IN ['Pune', 'Mumbai']

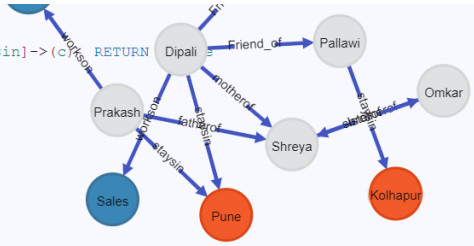
and (p)-[:staysin]->(c)

RETURN p.name

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

p.name
Dipali
Prakash
Meenal
Kaustubh

Query took 1227 ms and returned 4 rows. Result Details



4) List the names of people who are mothers.

→ match(p:Person),(pp:Person)

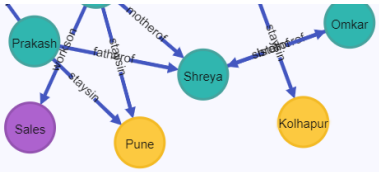
where (p)-[:motherof]->(pp)

return DISTINCT p.name

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

p.name
Dipali
Meenal

Query took 42 ms and returned 2 rows. Result Details



5) Display the names of people living in Mumbai.

→ match(p:Person),(c:City)

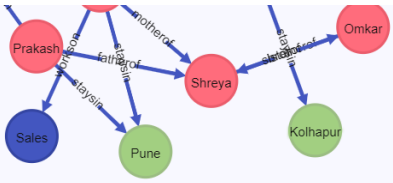
where c.name="Mumbai" and (p)-[:staysin]->(c)

return p.name

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

p.name
Meenal
Kaustubh

Query took 17 ms and returned 2 rows. Result Details



6) Display the nodes having age above 40.

→ match(p:Person)

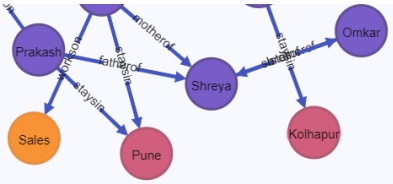
where p.age>40

return p.name

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

p.name
Pallawi
Kaustubh

Query took 54 ms and returned 2 rows. Result Details



Chapter 9 Shopping Mall

Model the following Clothing Brand information as a graph model, and answer the following queries using Cypher. Consider a Mall for clothing. This mall will include different sections for males, females and kids. Each section contains different types of apparels from different brands. There are many apparels with different designs, of each type. An apparel may be available in one or more standard sizes (S/M/L/XL/XXL). Sales staff is assigned for each section.

Node Creation: Section

```
Create(:Section{name:"Male"})
Create(:Section{name:"Female"})
Create(:Section{name:"Kid"})
```

Node Creation: Apparel

```
Create(:Apparel{name:"T-shirt"})
Create(:Apparel{name:"Shirt"})
Create(:Apparel{name:"Kurti"})
Create(:Apparel{name:"Jacket"})
```

Node Creation: Brand

```
Create(:Brand{name:"Adidas"})
Create(:Brand{name:"Myntra"})
```

Node Creation: Size

```
Create(:Size{name:"S"})
Create(:Size{name:"M"})
Create(:Size{name:"L"})
Create(:Size{name:"XL"})
Create(:Size{name:"XXL "})
```

Node Creation: Staff

```
Create(:Staff{name:"Dipali"})
Create(:Staff{name:"Prakash"})
Create(:Staff{name:"Pallawi"})
```

Node Creation: Design

```
create(:Design {name:"Full Sleeves"})
create(:Design {name:"Half Sleeves"})
```

Relationship Creation: assigned to

```
Match(ss:Staff),(s:Section)
where ss.name="Pallawi" and s.name="Female"
create (ss)-[:assignedto]->(s) return s,ss

Match(ss:Staff),(s:Section)
where ss.name="Prakash" and s.name="Male"
create (ss)-[:assignedto]->(s) return s,ss

Match(ss:Staff),(s:Section)
where ss.name="Dipali" and s.name="Kid"
create (ss)-[:assignedto]->(s) return s,ss
```

Relationship Creation: contains

```
Match (s:Section),(a:Apparel)
where s.name="Kid"  and a.name="T-shirt"
create (s)-[:contains]->(a) return s,a

Match (s:Section),(a:Apparel)
where s.name="Female"  and a.name="T-shirt"
create (s)-[:contains]->(a) return s,a

Match (s:Section),(a:Apparel)
where s.name="Female"  and a.name="Kurti"
create (s)-[:contains]->(a) return s,a

Match (s:Section),(a:Apparel)
where s.name="Female"  and a.name="Jacket"
create (s)-[:contains]->(a) return s,a
```

```
Match (s:Section),(a:Apparel)
where s.name="Male"  and a.name="Jacket"
create (s)-[:contains]->(a) return s,a

Match (s:Section),(a:Apparel)
where s.name="Male"  and a.name="Shirt"
create (s)-[:contains]->(a) return s,a

Match (s:Section),(a:Apparel)
where s.name="Male"  and a.name="T-shirt"
create (s)-[:contains]->(a) return s,a
```

Relationship Creation: from

```
Match (a:Apparel),(b:Brand)
where a.name="T-shirt" and b.name="Adidas"
create (a)-[:from]->(b) return b,a

Match (a:Apparel),(b:Brand)
where a.name="T-shirt" and b.name="Myntra"
create (a)-[:from]->(b) return b,a

Match (a:Apparel),(b:Brand)
where a.name="Shirt" and b.name="Adidas"
create (a)-[:from]->(b) return b,a

Match (a:Apparel),(b:Brand)
where a.name="Kurti" and b.name="Myntra"
create (a)-[:from]->(b) return b,a

Match (a:Apparel),(b:Brand)
where a.name="Jacket" and b.name="Myntra"
create (a)-[:from]->(b) return b,a
```

Relationship Creation: design of

```
Match (a:Apparel),(d:Design)
where a.name="Jacket" and d.name="Full Sleeves"
create (a)-[:design_of]->(d) return d,a

Match (a:Apparel),(d:Design)
where a.name="T-shirt" and d.name="Full Sleeves"
create (a)-[:design_of]->(d) return d,a

Match (a:Apparel),(d:Design)
where a.name="T-shirt" and d.name="Half Sleeves"
create (a)-[:design_of]->(d) return d,a

Match(a:Apparel),(d:Design)
where d.name="Full Sleeves" and a.name="Kurti"
create (a)-[:design_of]->(d)    return a,d
```

Relationship Creation: available in

```
Match (a:Apparel),(s:Size)
where a.name="T-shirt" and s.name="M"
create (a)-[:availablein]->(s) return s,a

Match (a:Apparel),(s:Size)
where a.name="T-shirt" and s.name="L"
create (a)-[:availablein]->(s) return s,a

Match (a:Apparel),(s:Size)
where a.name="Shirt" and s.name="L"
create (a)-[:availablein]->(s) return s,a

Match (a:Apparel),(s:Size)
where a.name="T-shirt" and s.name="S"
create (a)-[:availablein]->(s) return s,a
```

Match (a:Apparel),(s:Size)

where a.name="Kurti" and s.name="M"

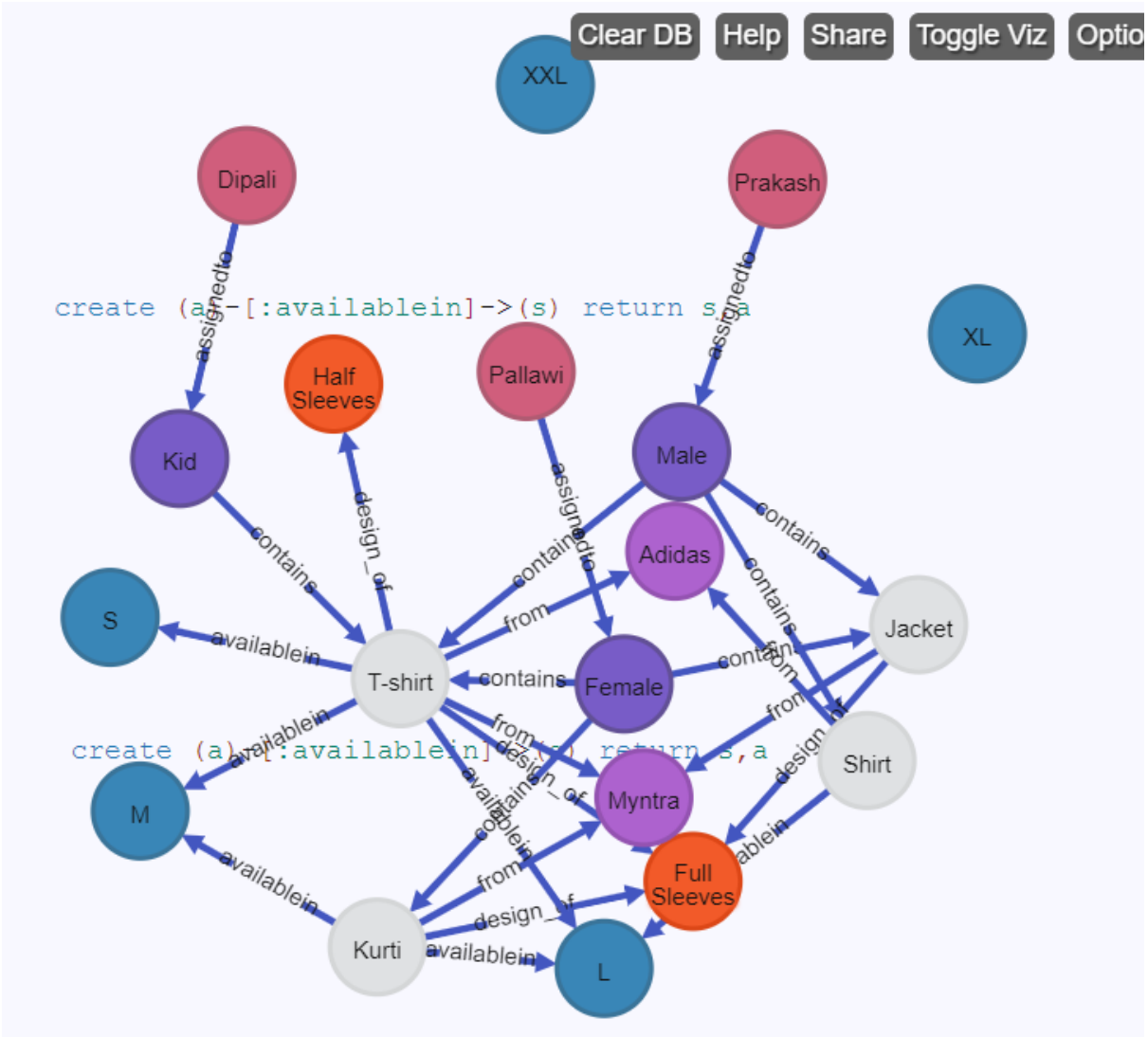
create (a)-[:availablein]->(s) return s,a

Match (a:Apparel),(s:Size)

where a.name="Kurti" and s.name="L"

create (a)-[:availablein]->(s) return s,a

Neo4j Model



Queries:

Query 1) List the different apparels type is female section

Ans→

Match(a:Apparel),(d:Design),(s:Section)

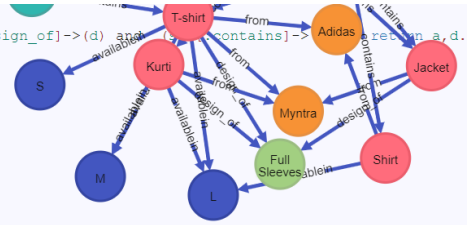
where s.name="Female" and

(a)-[:design_of]->(d) and (s)-[:contains]->(a) return a,d.name

```
Query:
Match(a:Apparel),(d:Design),(s:Section)  where s.name="Female"  and  (a)-[:design_of]->(d) and  (s)-[:contains]->(a)  return a,d.name
```

a	d.name
(35:Apparel {name:"T-shirt"})	Full Sleeves
(37:Apparel {name:"Kurti"})	Full Sleeves
(38:Apparel {name:"Jacket"})	Full Sleeves
(35:Apparel {name:"T-shirt"})	Half Sleeves

Query took 116 ms and returned 4 rows. Result Details



Query 2) List the names of sales staff in kids section

Ans→

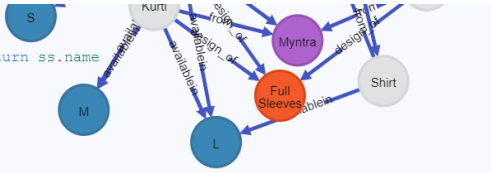
Match(ss:Staff),(s:Section)

where s.name="Kid" and (ss)-[:assignedto]->(s) return ss.name

```
Query:
Match(ss:Staff),(s:Section)  where s.name="Kid" and (ss)-[:assignedto]->(s) return ss.name
```

ss.name
Dipali

Query took 19 ms and returned 1 rows. Result Details



Query 3) List the standard sizes available for shirts in male section

Ans→

match(s:Section),(sz:Size), (a:Apparel)

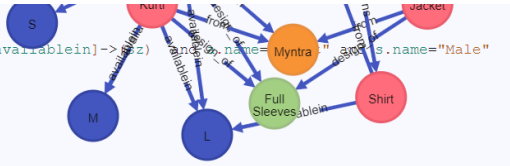
where (s)-[:contains]->(a) and (a)-[:availablein]->(sz)

and a.name="Shirt" and s.name="Male" return sz.name

```
Query:
match(s:Section),(sz:Size),  (a:Apparel)  where (s)-[:contains]->(a) and (a)-[:availablein]->(sz)  and a.name="Shirt" and s.name="Male"
return sz.name
```

sz.name
L

Query took 64 ms and returned 1 rows. Result Details



Query 4) List the brand having least number of apparels under it

Ans→

match (a:Apparel)-[:from]->(b:Brand)

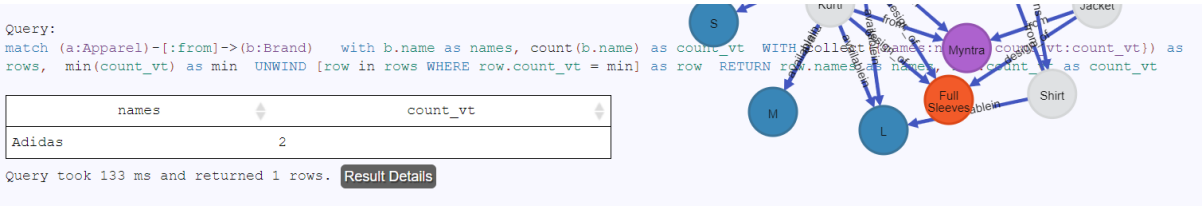
with b.name as names, count(b.name) as count_vt

WITH collect({names:names, count_vt:count_vt}) as rows,

min(count_vt) as min

UNWIND [row in rows WHERE row.count_vt = min] as row

RETURN row.names as names, row.count_vt as count_vt



Chapter 10 University

Model the following University information system as a graph model, and answer the following queries using Cypher. University has various departments like Mathematics, Zoology, 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.

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

3. Answer the Queries

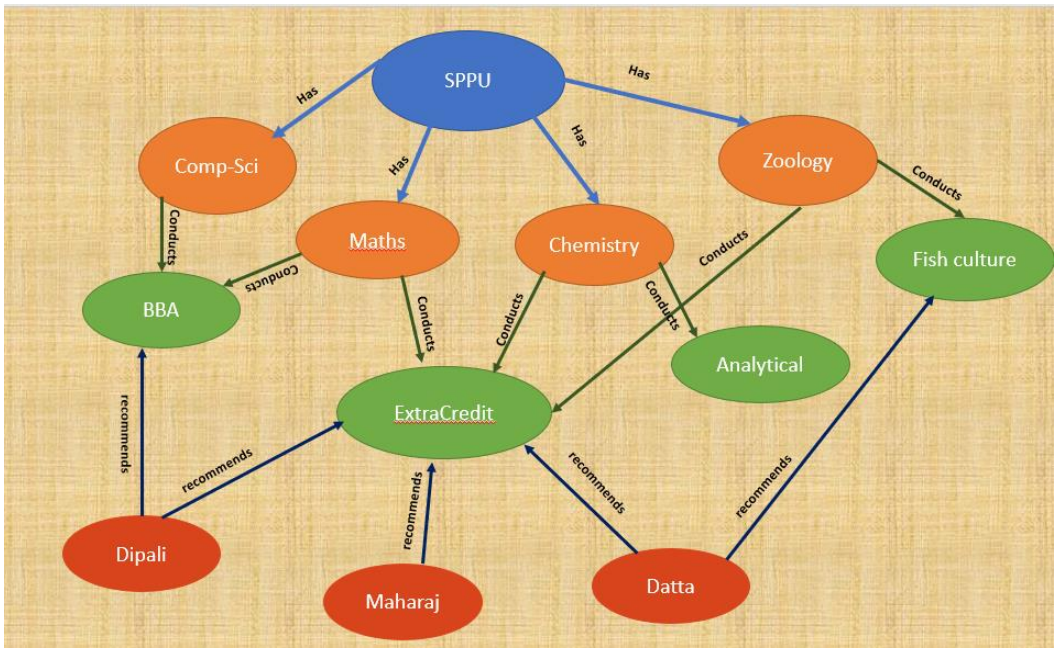
a. List the names of the courses provided by Chemistry Department.

b. List the details of all the departments in the university.

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

d. List the most recommended course in Computer Science Department.

[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: Has

```
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
```

Node Creation: Course

```
create (:Course{name:'ExtraCredit'})
create (:Course{name:'BBA'})
create (:Course{name:'Analytical'})
create (:Course{name:'Fish culture'})
```

Relationship Creation: conducts

```
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='Zoology' and c.name='Fish culture'
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

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

Node Creation: people

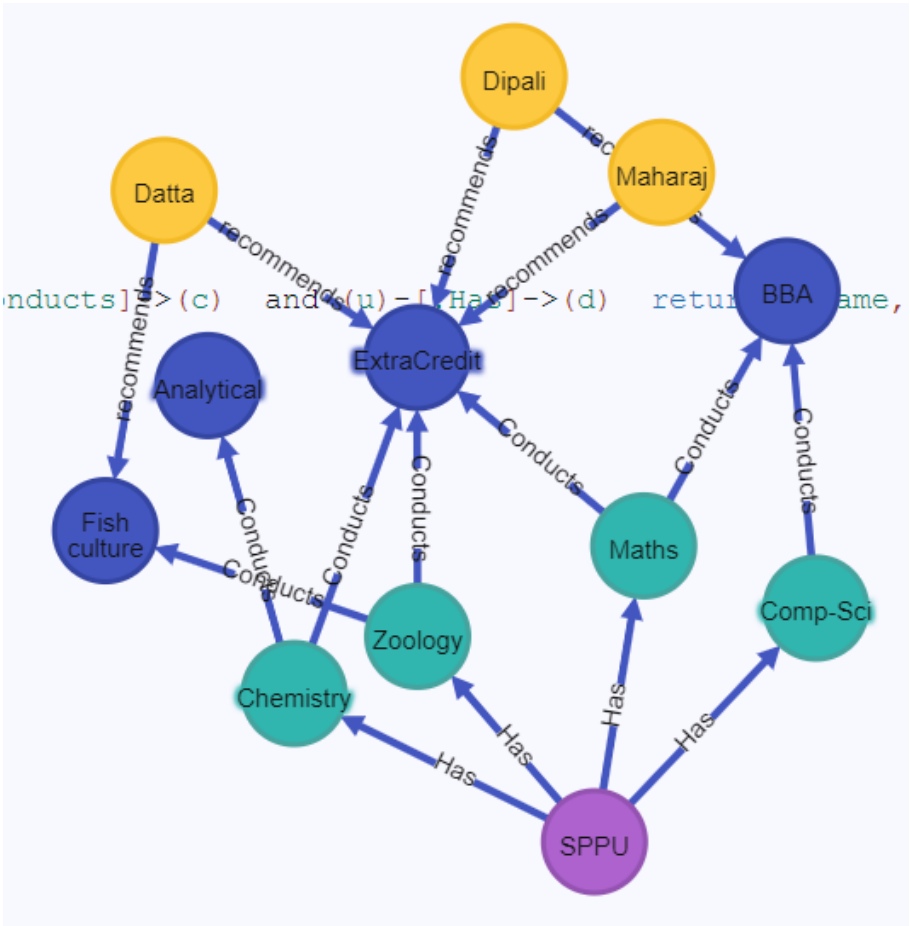
```
create (:People{name:'Dipali'})
create (:People{name:'Maharaj'})
create (:People{name:'Datta'})
```

Relationship Creation: recommends

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

```
MATCH (p:People), (c:Course)
WHERE p.name = "Datta" AND c.name = "Fish culture"
CREATE (p)-[:recommends ]->(c) RETURN p,c

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

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

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

Final Graph Created in Neo4j



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

output

d.name	c.name
Zoology	ExtraCredit
Maths	ExtraCredit
Chemistry	ExtraCredit
Comp-Sci	BBA
Maths	BBA
Chemistry	Analytical
Zoology	Fish culture

Query took 67 ms and returned 7 rows. Result Details

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.name

Output:

c.name
BBA

Query took 128 ms and returned 1 rows. Result Details

Query 3) courses run by chemistry department

→match(c:Course),(d:Department)

where d.name='Chemistry' and

(d)-[:Conducts]->(c) return c.name

Output:

c.name
ExtraCredit
Analytical

Query took 97 ms and returned 2 rows. Result Details

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

→ MATCH (p:People)-[:recommends]->(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 coursename, row.count_vt as noof_recomm

Output:

coursename	noof_recomm
ExtraCredit	3

Query took 687 ms and returned 1 rows. Result Details

Chapter 11 Vehicle

Model the following Automobile information system as a graph model, and answer the following queries using Cypher. Consider an Automobile industry manufacturing different types of vehicles like Heavy Vehicles, Light Vehicles, etc. A customer can buy one or more types of vehicle. A person can recommend or rate a vehicle type.

Node Creation: Industry

```
create( i:industry{name:"Meher Industries"})
```

Node Creation: Vehicle type

```
create( vt:VehicleType{name:"Light"})
return vt
create( vt:VehicleType{name:"Heavy"})
return vt
```

Relationship creation: manufactures

```
match (i:industry),(vt:VehicleType)
where i.name="Meher Industries" and vt.name="Light"
create (i)-[:manufactures]->(vt) return i,vt

match (i:industry),(vt:VehicleType)
where i.name="Meher Industries" and vt.name= "Heavy"
create (i)-[:manufactures]->(vt) return i,vt
```

Node Creation: Vehicle

```
create(:Vehicle{name:'Activa',color:'red',wheels:2,mode:'Manual',Mileage:48})
create(:Vehicle{name:'Pleasure',color:'white',wheels:2,mode:'Auto',Mileage:40})
create(:Vehicle{name:'HeroHonda',color:'blue',wheels:2,mode:'Auto',Mileage:60})
create(:Vehicle{name:'Deo',color:'red',wheels:2,mode:'Manual',Mileage:40})
create(:Vehicle{name:'Road Roller', color:'yellow', wheels:9,
engineworkson:'Dizel',Mileage:20})
```

```
create(:Vehicle{name:'Crane', color:'white', wheels:8,
engineworkson:'Dizel',capacity:'20ton',Mileage:20})
create(:Vehicle{name:'Dumper', color:'red',
wheels:10,engineworkson:'Dizel',capacity:'100ton',Mileage:20})
create(:Vehicle{name:'Truck',color:'red',wheels:10,mode:'Auto',Mileage:30})
```

Relationship creation: type of

```
match(v:Vehicle),(vt:VehicleType)
where v.name="Activa" and vt.name="Light"
create(v)-[:type_of]->(vt) return v,vt;

match(v:Vehicle),(vt:VehicleType)
where v.name="Pleasure" and vt.name="Light"
create(v)-[:type_of]->(vt) return v,vt;

match(v:Vehicle),(vt:VehicleType)
where v.name="HeroHonda" and vt.name="Light"
create(v)-[:type_of]->(vt) return v,vt;

match(v:Vehicle),(vt:VehicleType)
where v.name="Dumper" and vt.name="Heavy"
create(v)-[:type_of]->(vt) return v,vt;

match(v:Vehicle),(vt:VehicleType)
where v.name="Crane" and vt.name="Heavy"
create(v)-[:type_of]->(vt) return v,vt;

match(v:Vehicle),(vt:VehicleType)
where v.name="Road Roller" and vt.name="Heavy"
create(v)-[:type_of]->(vt) return v,vt;

match(v:Vehicle),(vt:VehicleType)
where v.name="Truck" and vt.name="Heavy"
create(v)-[:type_of]->(vt) return v,vt;
```

Node Creation: Customer

```
create(:customer{name:"Poonam"})
create(:customer{name:"Dipali"})
create(:customer{name:"Prakash"})
create(:customer{name:"Akshay"})
```

Relationship Creation: buy

```
match(c:customer),(v:Vehicle)
where c.name="Dipali" and v.name="Activa"
create(c)-[:buy]->(v) return c,v;

match(c:customer),(v:Vehicle)
where c.name="Prakash" and v.name="Crane"
create(c)-[:buy]->(v) return c,v;

match(c:customer),(v:Vehicle)
where c.name="Prakash" and v.name="Pleasure"
create(c)-[:buy]->(v) return c,v;

match(c:customer),(v:Vehicle)
where c.name="Akshay" and v.name="Activa"
create(c)-[:buy]->(v) return c,v;

match(c:customer),(v:Vehicle)
where c.name="Akshay" and v.name="Truck"
create(c)-[:buy]->(v) return c,v;

match(c:customer),(v:Vehicle)
where c.name="Poonam" and v.name="Truck"
create(c)-[:buy]->(v) return c,v;

match(c:customer),(v:Vehicle)
where c.name="Poonam" and v.name="Dumper"
```

```
create(c)-[:buy]->(v) return c,v;
match(c:customer),(v:Vehicle)
where c.name="Poonam" and v.name="Deo"
create(c)-[:buy]->(v) return c,v;
```

Node Creation: Person

```
create(:person{name:"Maharaj"})
create(:person{name:"Kusum"})
create(:person{name:"Gajanan"})
```

Node Creation: recommend

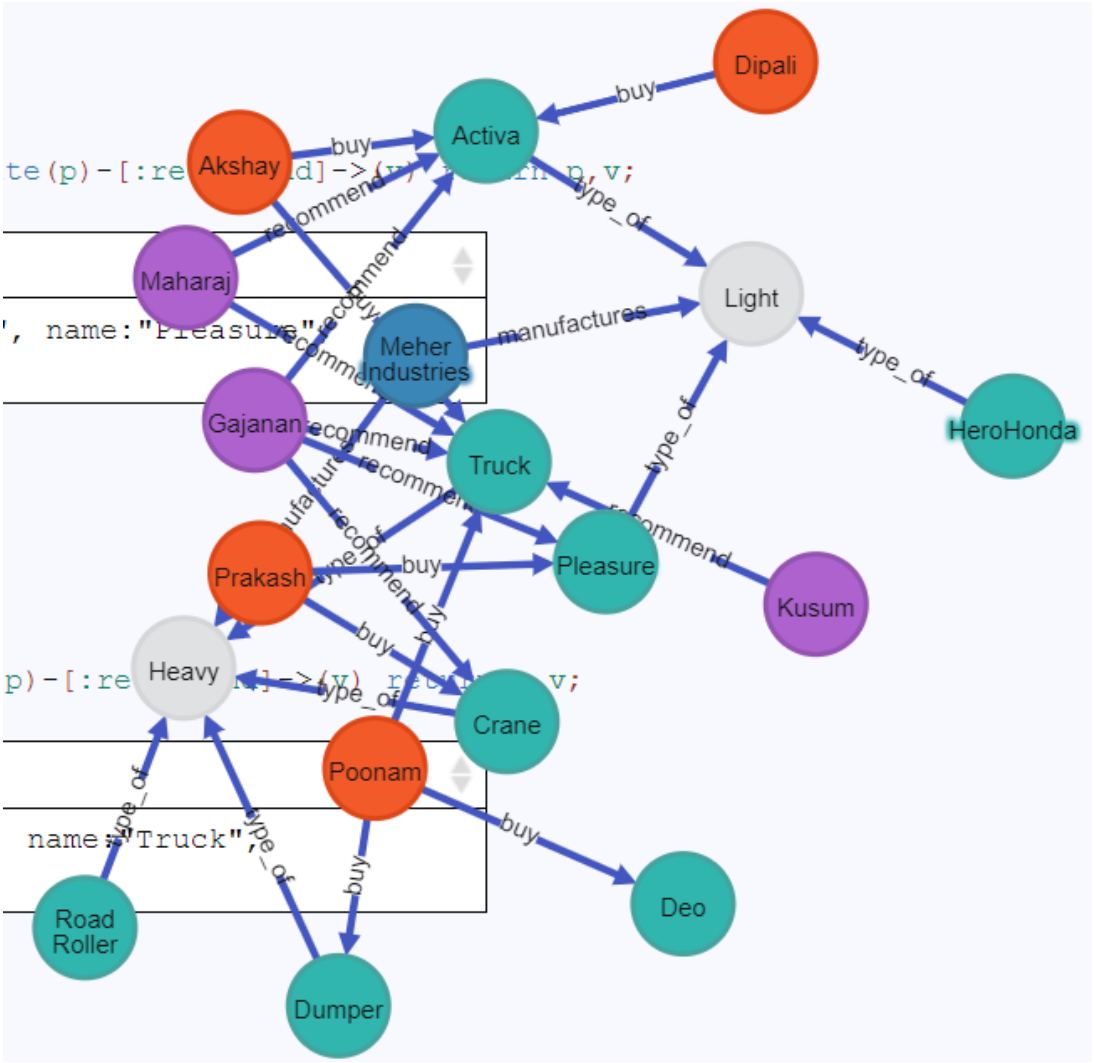
```
match(p:person),(v:Vehicle)
where p.name="Maharaj" and v.name="Activa"
create(p)-[:recommend]->(v) return p,v;

match(p:person),(v:Vehicle)
where p.name="Kusum" and v.name="Truck"
create(p)-[:recommend]->(v) return p,v;

match(p:person),(v:Vehicle)
where p.name="Maharaj" and v.name="Truck"
create(p)-[:recommend]->(v) return p,v;

match(p:person),(v:Vehicle)
where p.name="Gajanan" and v.name="Activa"
create(p)-[:recommend]->(v) return p,v;

match(p:person),(v:Vehicle)
where p.name="Gajanan" and v.name="Crane"
create(p)-[:recommend]->(v) return p,v;

match(p:person),(v:Vehicle)
where p.name="Gajanan" and v.name="Pleasure"
create(p)-[:recommend]->(v) return p,v;
```

match(p:person),(v:Vehicle)

where p.name="Gajanan" and v.name="'Truck"

create(p)-[:recommend]->(v) return p,v;

Model in Neo4j:



<u>Queries:</u>

<u>Query1) List the characteristics of heavy vehicle types.</u>

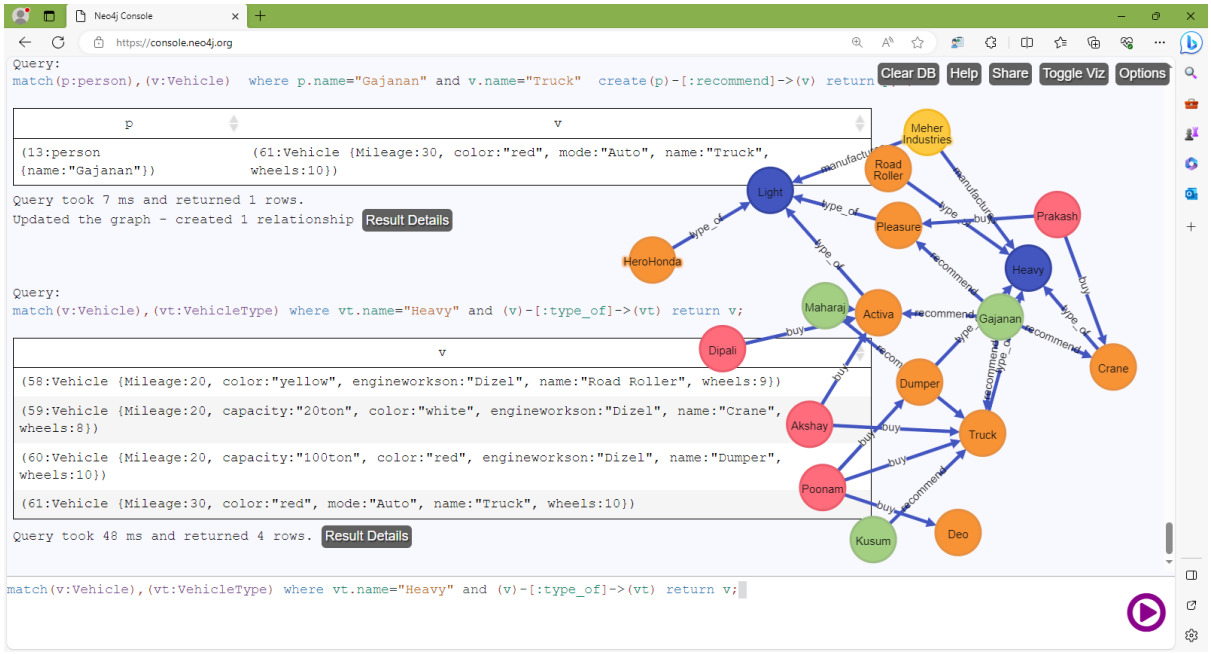
Ans→

match(v:Vehicle),(vt:VehicleType)

where vt.name="Heavy" and

(v)-[:type_of]->(vt)

return v;



Query 2) Query:List the name of customers who bought a light vehicle.

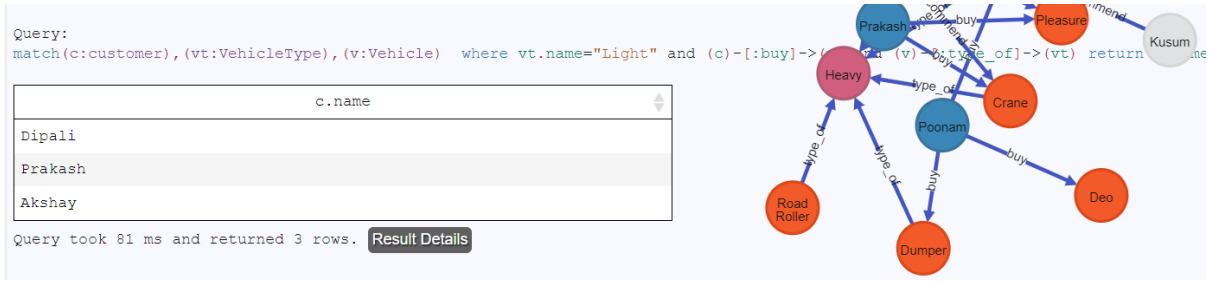
match(c:customer),(vt:VehicleType),(v:Vehicle)

where vt.name="Light" and

(c)-[:buy]->(v) and

(v)-[:type_of]->(vt)

return c.name



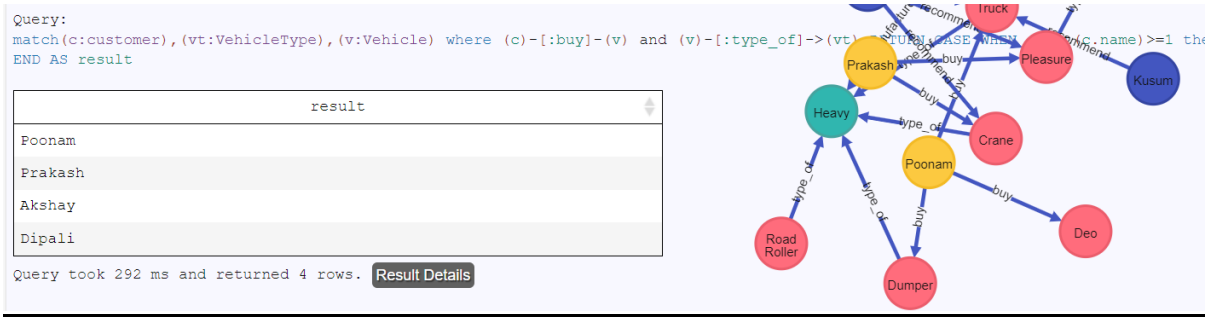
Query 3): List the customers who bought more than one type of vehicle

match(c:customer),(vt:VehicleType),(v:Vehicle)

where (c)-[:buy]-(v) and

(v)-[:type_of]->(vt)

RETURN CASE WHEN count(c.name)>=1 then c.name END AS result



Query 4): List the most recommended vehicle type

MATCH (p:person)-[:recommend]->(v:Vehicle),

(v:Vehicle)-[:type_of]->(vt:VehicleType)

with vt.name as names,

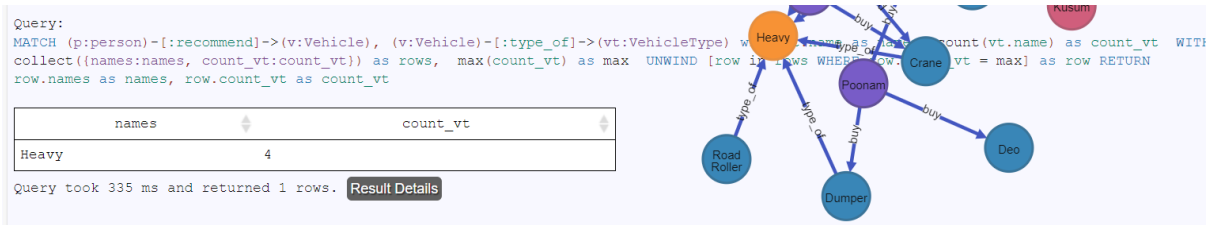
count(vt.name) as count_vt

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



Chapter 12 Import Export

There are countries which import and export products to each other. Products are produced across different states in a country . Production of the products is measured in %. A product can be exported if its production exceeds 60%. A product needs to be imported if its consumption percentage is more than its production percentage in a country.

Node Creation: Country

```
create(: Country{name:"India"})
 create(: Country{name:"UAE"})
create(: Country{name:"US"})
create(: Country{name:"Germany"})
create(:Country{name:"AAAA"})
```

node creation: state

```
create(:State{name:"Maharashtra"})
create(:State{name:"Aasam"})
create(:State{name:"Dubai"})
create(:State{name:"Canada"})
create(:State{name:"Mexico"})
create(:State{name:"Berlin"})
create(:State{name:"SSSS"})
```

relationship creation: has

```
match(c:Country),(s:State) where c.name="India" and s.name="Maharashtra"
create (c)-[:Has]->(s) return c,s

match(c:Country),(s:State) where c.name="India" and s.name="Aasam"
create (c)-[:Has]->(s) return c,s

match(c:Country),(s:State) where c.name="UAE" and s.name="Dubai"
create (c)-[:Has]->(s) return c,s
```

```
match(c:Country),(s:State) where c.name="Germany" and s.name="Mexico"
create (c)-[:Has]->(s) return c,s

match(c:Country),(s:State) where c.name="Germany" and s.name="Canada"
create (c)-[:Has]->(s) return c,s

match(c:Country),(s:State) where c.name="US" and s.name="Berlin"
create (c)-[:Has]->(s) return c,s

match(c:Country),(s:State) where c.name="AAAA" and s.name="Nepal"
create (c)-[:Has]->(s) return c,s
```

node creation: product

```
create(:Product{name:"Oil"})
create(:Product{name:"Sugarcane"})
create(:Product{name:"Rice"})
create(:Product{name:"Lentils"})
```

relationship creation: across

```
match(p:Product),(s:State) where p.name="Oil" and s.name="Dubai" create(p)-
[:produced_across {pper:74,cper:50}]->(s) return p,s

match(p:Product),(s:State) where p.name="Rice" and s.name="Maharashtra" create(p)-
[:produced_across{pper:80,cper:45}]->(s) return p,s
match(p:Product),(s:State) where p.name="Lentils" and s.name="Canada" create(p)-
[:produced_across{pper:40,cper:80}]->(s) return p,s

match(p:Product),(s:State) where p.name="Sugarcane" and s.name="Mexico" create(p)-
[:produced_across{pper:50,cper:50}]->(s) return p,s

match(p:Product),(s:State) where p.name="Oil" and s.name="Aasam" create(p)-
[:produced_across {pper:59,cper:79}]->(s) return p,s
```

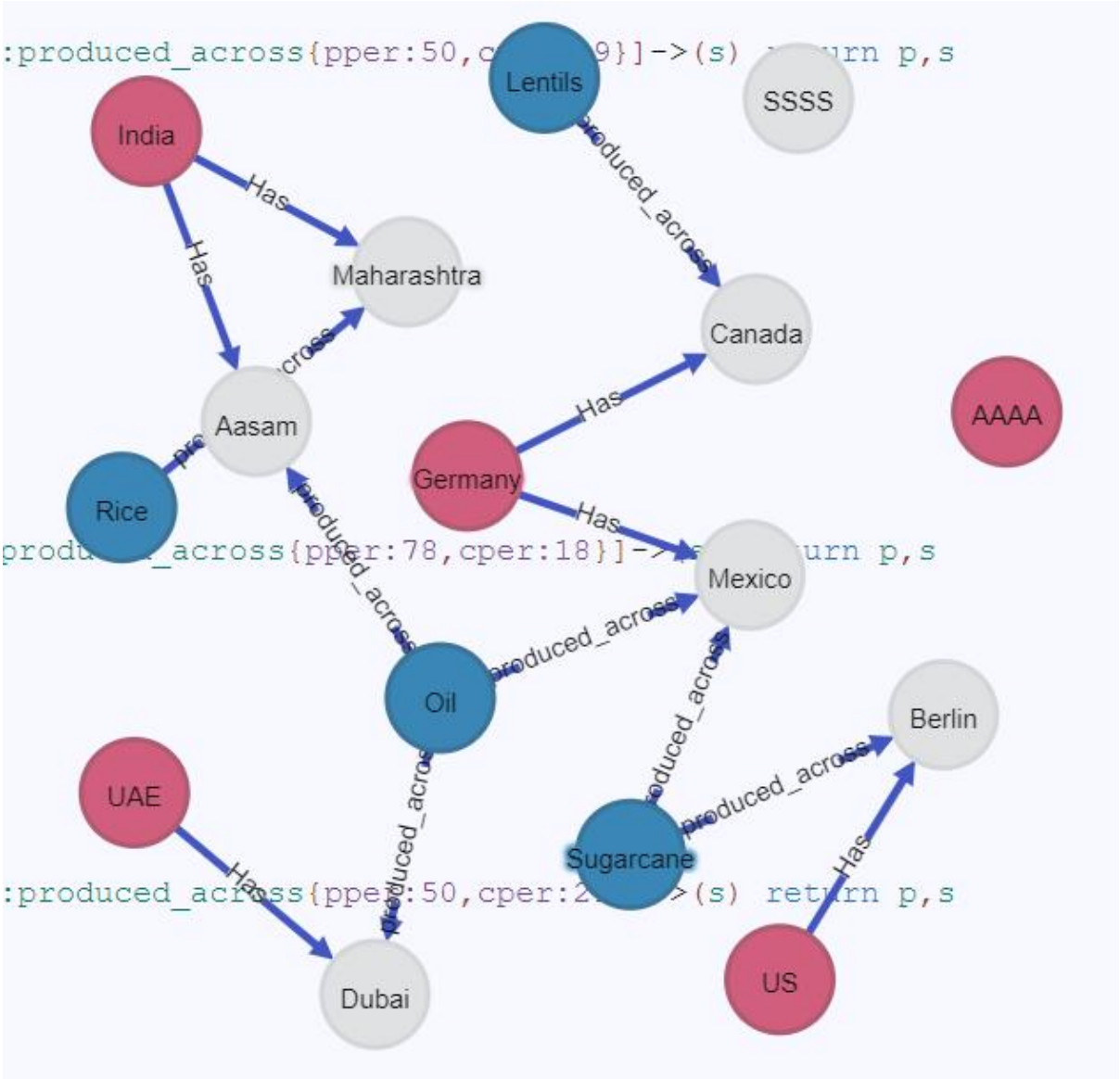
match(p:Product),(s:State) where p.name="Sugarcane" and s.name="Berlin" create(p)-[:produced_across{pper:72,cper:50}]->(s) return p,s

match(p:Product),(s:State) where p.name="Oil" and s.name="Mexico" create(p)-[:produced_across{pper:50,cper:89}]->(s) return p,s

match(p:Product),(s:State) where p.name="Oil" and s.name="Nepal" create(p)-[:produced_across{pper:78,cper:18}]->(s) return p,s

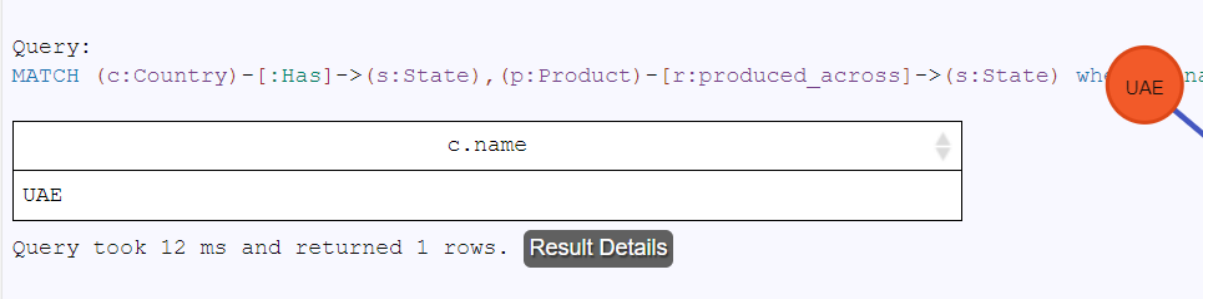
match(p:Product),(s:State) where p.name="Rice" and s.name="Nepal" create(p)-[:produced_across{pper:50,cper:29}]->(s) return p,s

graph:



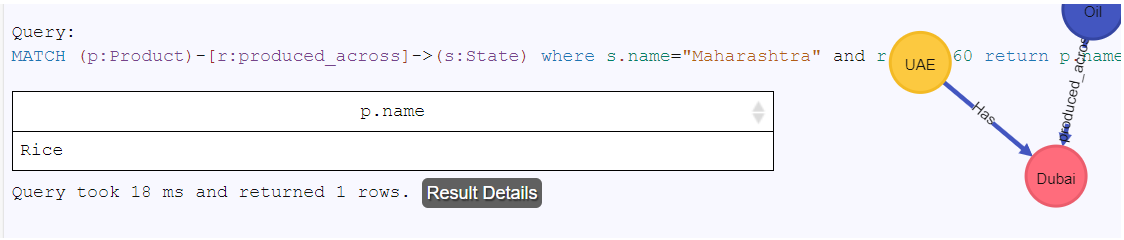
1. List the countries that export oil

MATCH (c:Country)-[:Has]->(s:State),(p:Product)-[r:produced_across]->(s:State) where p.name="Oil" and r.pper>60 return c.name



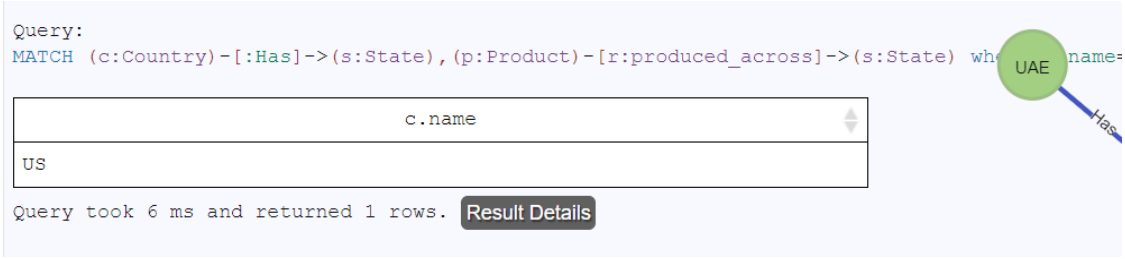
2. List the products produced in maharashtra

MATCH (p:Product)-[r:produced_across]->(s:State) where s.name="Maharashtra" and r.pper>60 return p.name



3. List the countries that produce more than 70% sugarcane

MATCH (c:Country)-[:Has]->(s:State),(p:Product)-[r:produced_across]->(s:State) where p.name="Sugarcane" and r.pper>70 return c.name



Bibliography:

1. https://neo4j.com/
2. https://stackoverflow.com/
3. https://www.taylorfrancis.com/chapters/edit/10.1201/9781003358596-26/databases-machine-learning-journey-sql-nosql-dipali-meher-baljeet-kaur-alaknanda-pawar-sheetal-parekh
4. https://neo4j.com/case-studies/
5. https://www.ijeast.com/papers/216-219,%20Tesma0802,IJEAST.pdf
6. https://neo4j.com/docs/java-reference/current/
7. https://neo4j.com/docs/
8. https://www.researchgate.net/publication/307180380_Literature_review_about_Neo4j_graph_database_as_a_feasible_alternative_for_replacing_RDBMS
9. https://neo4j.com/graphgists/interpreting-citation-patterns-in-academic-publications-a-research-aid/
10. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0207595

Handbook of Case Studies in Neo4j

ISBN: 978-93-88901-93-2

About Author

Dr. Dipali P. Meher
Assistant Professor and M.Sc. CS Course Coordinator,
Department of Computer Science,
Modern College of Arts, Science and Commerce, Ganeshkhind, Pune 16

Dr. Dipali P. Meher has completed Ph.D. in from the Bharati Vidyapeeth (Deemed to be University) , Maharashtra, INDIA. She has also qualified NET exam from UGC. She has done M.Phil. (Computer Science) and obtained First Class in M.C.S. (Computer Science), B.C.S.(Computer Science) degree from Pune University. She is working as an Assistant Professor and M.Sc. CS Course Coordinator at the Department of Computer Science, Modern College of Arts, Science and Commerce, Ganeshkhind, Pune 16. Teaching is her passion and she has 20 years of experience. She has Received recognition as Post Graduate teacher from Savitribai Phule Pune University. She has worked as a Research paper Reviewer for the journal International Journal of Engineering Research and Technology. She has 12 good research publications in International Journals and conferences and author of 8 books.

Bhumi Publishing

ISBN 978-93-88901-93-2
9 789388 901932