Database Technologies: Neo4j Practical Assignment 1

Create the following databases as graph models. Visualize the models after creation, Return properties of nodes, Return the nodes labels, Return the relationships with its properties.
NB: You may assume and add more labels , relationships, properties to the graphs

1. Create a Library database,
There are individual books, readers, and authors that are present in the library data model.. A minimal set of labels are as follows:

Book : This label includes all the books
Person : This label includes authors, translators, reviewers, Readers, Suppliers and so on
Publisher : This label includes the publishers of books in the database

A set of basic relationships are as follows:
PublishedBy : This relationship is used to specify that a book was published by a publisher **Votes**: This relationship describes the relation between a user and a book, for example, how a book was rated by a user.
ReviewedBy : This relationship is used to specify that a book was reviewed and remarked by a user.
TranslatedBy : This relationship is used to specify that a book was translated to a language by a user.
IssuedBy : This relationship is used to specify that a book was issued by a user.
ReturnedBy : This relationship is used to specify that a book was returned by a user

Every book has the following properties:
Title : This is the title of the book in string format
Tags : This is an array of string tags useful for searching through the database based on topic, arguments, geographic regions, languages, and so on
Status : the book status , specifying whether its issued or in library.
Condition: book condition, new or old
Cost : Cost of book
Type : book is a Novel, Journal, suspense thriller etc

```
#View All nodes, labels, Relationships etc.
---> match (n) return n

#Delete All nodes, labels, Relationships etc.
---> MATCH (n) DETACH DELETE n

@CREATE (tinker:Book{title:'Bayari',tag: [ 'Social Issues','Maharashtra'],
 published:1988,cost:550, type:'Novel'})

#Delete All nodes, labels, Relationships where Book="Bayari"
--->
match (n:Book{title:'Bayari'}) Detach delete n



CREATE(pk:Publisher{name:'PK',city:'Pune'})

CREATE (john:Author{name:'John Le Carre', born:'19-10-1932'})
CREATE (graham:Author{name:'Graham Greene',born:'02-10-1904',died:'02-04-
1991'})
```

```
CREATE (tinker:Book{title:"Tinker Tailor Soldier Spy",
tag:['English', 'Japanies'], status:'not Issued', condition:'New',
published:1974, cost:350, type:'Novel'})
CREATE (our:Book{title:'Our Man in Havana',
tag:['Ameriaca','English','Korian'], status:'Issued',condition:'new',
published:1958, cost:250, type:'suspense thriller'})

CREATE (lan:Reader:Author{name:'Lan'})
CREATE (alan:Reader{name:'Alan'})
CREATE(clay:Reader{name:'Clay'})
CREATE(han:Reader{name:'Hanahha Baker'})

CREATE(Jassica:Auther{name:'Jassica'})

CREATE  (our)-[:PUBLISHED_BY]->(pk),
      (Jassica)<-[:TRANSLATED_BY]-(tinker),
      (our)-[:ISSUED_BY]->(clay),
      (tinker)-[:REVIEWED_BY]->(pk),
      (han)-[:VOTES{stars:4}]->(our),
      (tinker)-[:PUBLISHED_BY]->(pk),
      (john)-[:WROTE]->(tinker),
      (alan)-[:RECOMMENDED{date:'05-07-2011'}]->(tinker),
      (lan)-[:RECOMMENDED{date:'09-09-2011'}]->(tinker),
      (lan)-[:RECOMMENDED{date:'03-02-2011'}]->(our),
      (graham)-[:WROTE]->(our)

CREATE(b:Book{title:"Bayari",
tag:['English', 'Marathi'], status:'not Issued', condition:'New',
published:1999, cost:333, type:'Novel'})
match (lan:Reader{name:'Lan'})
create (b)-[:RECOMMENDED]->(lan)
return b,lan

match (:Reader{Name:"Lan"})-[r:RECOMMENDED]->(:Book{title="Tinker Tailor
Soldier Spy"})
detach delete r

match (b:Book),(r:Reader)
where b.title="Our Man in Havana" and r.name="Hanahha Baker"
create(b)-[i:ISSUED_BY]->(r)
return b,r

match (b:Book),(r:Reader)
where b.title="Tinker Tailor Soldier Spy" and r.name:"Clay"
create(b)-[i:ISSUED_BY]->(r)
return b,r

CREATE (sane:Author{name:'sane guruji', born:'09-10-1942', city :"Satara"})

match(b:Book),(a:Author)
where b.title="Bayari" and a.name="sane guruji"
create (a)-[:WROTE]->(b)
return a,b

match (p:Publisher)<-[r:PUBLISHED_BY]-(b:Book)<-[rr:WROTE]-(a:Author)
return r,rr
```

2. 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.

```
CREATE(pk:Artist:Song_Author{Name:'PK', Age:20, followers:'50M'})
CREATE(bantai:Artist:Song_Author{Name:'Emiway Bantai', Age:26,
followers:'5M'})
CREATE(guru:Artist:Song_Author{Name:'Guru', Age:27, followers:'12M'})
CREATE(raf:Artist:Song_Author{Name:'Raftaar', Age:30, followers:'4M'})
CREATE(divine:Artist:Song_Author{Name:'Divine', Age:31, followers:'14M'})
CREATE(neha:Artist:Song_Author{Name:'Neha Kakkar', Age:29, followers:'3M'})

CREATE(hard:Song{Name:'Bohot Hard', likes:'40M'})
CREATE(gully:Song{Name:'Mere Gully Main', likes:'12M'})
CREATE(azadi:Song{Name:'Azadi', likes:'7M'})
CREATE(asli:Song{Name:'Asli',likes:'8M'})
CREATE(gabru:Song{Name:'High Rated Gabru', likes:'10M'})
CREATE(ladki:Song{Name:'Ladki Marwake Marke Maneggii', likes:'2.5M'})
CREATE(machayenge:Song{Name:'Machayenge', likes:'8M'})
CREATE(chull:Song{Name:'Kar Gayi Chull', likes:'5M'})

CREATE(arijit:Song_Author{Name:'Arijit Singh',No_songs:50})
CREATE(tony:Song_Author{Name:'Tony Kakkar',No_songs:112})

CREATE(coke:Recording_company{Name:'Coke Studio'})
CREATE(zee:Recording_company:Recoding_studio{Name:'Zee Studio'})

CREATE  (pk)-[:PERFORMS]->(hard),
      (gabru)-[:WRITTEN_BY]->(bantai),
      (bantai)-[:PERFORMS]->(machayenge)-[:WRITTEN_BY]->(bantai),
      (guru)-[:PERFORMS]->(gabru)-[:WRITTEN_BY]->(arijit),
      (raf)-[:PERFORMS]->(ladki)-[:WRITTEN_BY]->(raf),
      (divine)-[:PERFORMS]->(gully)-[:WRITTEN_BY]->(divine),
      (divine)-[:PERFORMS]->(azadi)-[:WRITTEN_BY]->(divine),
      (neha)-[:PERFORMS]->(chull),
      (asli)-[:RECORDED_IN]->(zee)-[:MANAGED_BY]->(zee)-[:Finances]->(hard),
      (gabru)-[:RECORDED_IN]->(coke)-[:MANAGED_BY]->(zee),
      (ladki)-[:RECORDED_IN]->(zee)-[:MANAGED_BY]->(coke)-[:Finances]-
>(chull),
      (azadi)-[:RECORDED_IN]->(coke)-[:MANAGED_BY]->(coke),
      (neha)-[:FOLLOWS]->(arijit)-[:FOLLOWS]->(guru)-[:FOLLOWS]->(raf)-
[:FOLLOWS]->(tony)-[:FOLLOWS]->(pk)
```

3. Consider an Employee database,
with a minimal set of labels as follows Employee: denotes a person as an employee of the organization Department: denotes the different departments, in which employees work. Skillset: A list of skills acquired by an employee

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

-->

```
CREATE(harry:Employee {Name:'Harry', age:29, Qualification:['MCS','BCS'],
Experience:8})
CREATE(pashya:Employee {Name:'Pashya', age:30,
Qualification:['MCA','BCA','MSCIT'],Experience:8})
CREATE(bablu:Employee {Name:'Bablu', age:28,
Qualification:['B.Tech','MSCIT'],Experience:5})
CREATE(monu:Employee {Name:'Monu', age:26,
Qualification:['B.Tech','M.Tech'],Experience:3})
CREATE(babu:Employee {Name:'Babu', age:32,
Qualification:['M.Tech','B.Tech','MSCIT','MCS','BCS'],Experience:10})
CREATE(nandu:Employee {Name:'Nandu', age:34,
Qualification:['B.Tech','BCS','MSCIT'],Experience:4})

CREATE(it:Department{Name:'IT',no_of_Emp:5})
CREATE(bpo:Department{Name:'BPO',no_of_Emp:5})
CREATE(cbo:Department{Name:'CBO',no_of_Emp:5})
CREATE(cmn:Department{Name:'TeleCommunication',no_of_Emp:5})

CREATE(vgd:Skillset{skills:['Fluent Communication','Leadership
Qualities','Optimistic']})
CREATE(bet:Skillset{skills:['Good Communication','Java Devloper']})
CREATE(gd:Skillset{skills:['Leadership Qualities','Optimistic','Finnance
Matser']})
CREATE(av:Skillset{skills:['Fluent Communication']})

CREATE(sg:Projects{Name:'SG Website Design', TimeSpan:'30day', clinet:'SG
Architecture'})
CREATE(food:Projects{Name:'Food Deliver app', TimeSpan:'35day',
clinet:'Hydrabaad Biryanni'})
CREATE(location:Projects{Name:'Location Finder', TimeSpan:'45day',
clinet:'AI Location Developer'})
CREATE(ecom:Projects{Name:'Ecommerce Website Design', TimeSpan:'50day',
clinet:'Eco-market'})
CREATE(tata:Projects{Name:'Tata Sky', TimeSpan:'20day', clinet:'SkyVoice
India'})
CREATE(out:Projects{Name:'Out Bound Process', TimeSpan:'90day', clinet:'Ruby
Max'})

CREATE  (harry)-[:Works_in]->(it),
      (pashya)-[:Works_in]->(bpo),
      (bablu)-[:Works_in]->(cbo),
      (monu)-[:Works_in]->(cmn),
      (nandu)-[:Works_in]->(it),
      (harry)-[:Has_acquired]->(vgd),
      (pashya)-[:Has_acquired]->(bet),
```

```
(bablu)-[:Has_acquired]->(gd),
(monu)-[:Has_acquired]->(vgd),
(nandu)-[:Has_acquired]->(bet),
      (harry)-[:Has_acquired]->(av),
(harry)-[:Assigned_to]->(sg),
(nandu)-[:Assigned_to]->(food),
(bablu)-[:Assigned_to]->(ecom),
(monu)-[:Assigned_to]->(sg),
(pashya)-[:Assigned_to]->(location),
(harry)-[:Assigned_to]->(out),
(bablu)-[:Assigned_to]->(out),
(out)-[:Controlled_by]->(it),
(location)-[:Controlled_by]->(cbo),
(sg)-[:Controlled_by]->(it),
(ecom)-[:Controlled_by]->(bpo),
(food)-[:Controlled_by]->(cmn),
(nandu)-[:Project_manager]->(sg),
(bablu)-[:Project_manager]->(food),
(monu)-[:Project_manager]->(ecom),
(pashya)-[:Project_manager]->(location),
(babu)-[:Project_manager]->(tata),
(harry)-[:Project_manager]->(out)
```

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

```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome
to the Real World'})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})
CREATE (LanaW:Director{name:'Lana Wachowski', born:1965})
CREATE (JoelS {name:'Joel Silver', born:1952})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrix),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrix),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrix),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrix),
  (LillyW)-[:DIRECTED]->(TheMatrix),
  (LanaW)-[:DIRECTED]->(TheMatrix),
  (JoelS)-[:PRODUCED]->(TheMatrix)

CREATE (Emil:Person {name:"Emil Eifrem", born:1978})
CREATE (Emil)-[:ACTED_IN {roles:["Emil"]}]->(TheMatrix)

CREATE (TheMatrixReloaded:Movie {title:'The Matrix Reloaded', released:2003,
tagline:'Free your mind'})
CREATE
  (Keanu)-[:ACTED_IN {roles:['Neo']}]->(TheMatrixReloaded),
  (Carrie)-[:ACTED_IN {roles:['Trinity']}]->(TheMatrixReloaded),
  (Laurence)-[:ACTED_IN {roles:['Morpheus']}]->(TheMatrixReloaded),
  (Hugo)-[:ACTED_IN {roles:['Agent Smith']}]->(TheMatrixReloaded),
  (LillyW)-[:DIRECTED]->(TheMatrixReloaded),
```

```
  (LanaW)-[:DIRECTED]->(TheMatrixReloaded),
  (JoelS)-[:PRODUCED]->(TheMatrixReloaded)

CREATE (AngelaScope:Person {name:'Angela Scope'})
CREATE (JessicaThompson:Person {name:'Jessica Thompson'})

CREATE
  (JessicaThompson)-[:REVIEWED {summary:'An amazing journey', rating:95}]-
>(TheMatrixReloaded),
  (JessicaThompson)-[:REVIEWED {summary:'Silly, but fun', rating:65}]-
>(TheMatrix),
  (AngelaScope)-[:REVIEWED {summary:'Pretty funny at times', rating:62}]-
>(TheMatrixReloaded)

  Match (n) detach delete n
```

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

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

CREATE(j:person{name:"john",yoj:"2004",birthd+ay:"1996"}),(a:person{name:"am
an",birthday:"1996",yoj:"2005"}),(b:person{name:"bunny",birthday:"1995",yoj:
"2006"}),(e:affiliation{name:"facebook"}),
(g:group{name:"group"}),(t:timeline{name:"timeline"}),(s:story{name:"himacha
lstory"}),(m:message{name:"newmessage"}),(g1:group{name:"g1"}),(g2:group{nam
e:"g2"}),(g3:group{name:"g3"})

match (b:person),(s:story) where b.name ="bunny" and s.name =
"himachalstory"  create(s)-[r3:refers_to]-> (b)
match (m:message),(t:timeline) where m.name ="newmessage" and t.name =
"timeline"  create(t)-[r4:contains]-> (m)
match (s:story),(t:timeline) where s.name ="himachalstory" and t.name =
"timeline"  create(t)-[r4:reference_for]-> (s)
match (j:person),(t:timeline) where j.name ="john" and t.name = "timeline"
create(j)-[r4:creates]-> (t)
match (j:person),(s:story) where j.name ="john" and s.name = "himachalstory"
create(s)-[r3:refers_to]-> (j)
match (j:person),(s:story) where j.name ="john" and s.name = "himachalstory"
create(j)-[r3:create]-> (s)
match (a:person),(e:affiliation) where a.name ="aman" and e.name =
"facebook"  create(a)-[r1:affiliated_to]->(e)
match (j:person),(e:affiliation) where j.name ="john" and e.name =
"facebook"  create(j)-[r1:affiliated_to]->(e)
match (b:person),(e:affiliation) where b.name ="bunny" and e.name =
"facebook"  create(b)-[r1:affiliated_to]->(e)
match (j:person),(b:person) where j.name ="john" and b.name = "bunny"
create(j)-[r1:friend_of]-> (b)
match(j:person),(g1:group) create(j)-[r2:belongs_to]->(g1)
```

Database Technologies: Neo4j Practical Assignment 2 Simple Queries.

1. Library Database :

a) List all people, who have issued a book "Our Man in Havana".

```
-->MATCH (b:Book)-[r:ISSUED_BY]->(rd:Reader)
WHERE b.title='Our Man in Havana'
RETURN b,r,rd
```

b) Count the number of people who have read "Tinker Tailor Soldier Spy" .

-->[Note : Count this query doesn't give output for both conditions Issued_by and RECOMMENDED (i.e. both are readers)]

```
MATCH (a:Reader)-[r:RECOMMENDED]->(b:Book)
WHERE b.title="Tinker Tailor Soldier Spy"
RETURN COUNT(a)

MATCH (a:Reader)-[r:ISSUED_BY]->(b:Book)
WHERE b.title="Our Man in Havana"
RETURN COUNT(a)
```

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

```
-->[Note: These query is not running.]

MATCH (clay:Reader{name:'Clay'})
SET clay.No_of_Issued=4
RETURN clay
```

d) List the names of publishers from pune city.

```
-->match(p:Publisher{city:'Pune'})
return p.name
```

2.Song Database:

a) List the names of songs written by ":Emiway Bantai"

```
-->Match(s:Song)-[r:WRITTEN_BY]->(a:Song_Author)
where a.Name='Emiway Bantai'
return s.Name
```

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

```
-->Match(rec:Recording_company)-[r:Finances]->(s:Song)
where s.Name='Kar Gayi Chull'
return rec,s,r
```

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

```
-->MATCH (a:Artist)-[r:PERFORMS]->(s:Song)
WHERE s.Name='Bohot Hard'
RETURN a, s,r
```

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

```
-->MATCH (s:Song)-[r:RECORDED_IN]->(rec:Recoding_studio)
WHERE rec.Name='Zee Studio'
RETURN s,rec,r
```

3. Employee Database:

a) List the name of employees in department" IT".

```
-->match (e:Employee)-[:Works_in]->(:Department{Name:"IT"})
return  e.Name
```

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

```
-->match  (d:Department{Name:'IT'})<-[:Controlled_by]-(p:Projects)
return d,p
```

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

```
-->[Note : Not solved yet]
match (:Employee)-[:Works_in]->(d:Department)--((e:Employee))
return d,COUNT(e)
```

d) List the skillset for an employee "Harry"

```
-->MATCH (e:Employee{Name:"Harry"})-[:Has_acquired]->(s:Skillset)
return s,e
```

4. Movie Database:

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

```
-->MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
where m.title='The Matrix'
return p.name
```

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

```
-->match (p:Person)-[:REVIEWED]->(m:Movie),  (:Movie)<-[:REVIEWED]-(p)
return p.name,m.title
```

c) Find all actors that acted in a movie together after 2000 and return the actor names and movie node .

```
-->MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE m.released>1998
RETURN p.name, m.title
```

d) Find all movies produced by " Joel Silver".

```
-->MATCH (p:Person)-[:PRODUCED]->(m:Movie)
where p.name='Joel Silver'
Return m.title
```

5. Social Network Database:

a) List out the affiliations of John.

```
-->match(j:person{name:"john"})-[r1:affiliated_to]->(n) return n.name
```

b) Find all friends of "John", along with the year, since when john knows them.

```
-->match(j:person{name:"john"})-[r1:friend_of]->(n) return n.name,n.yoj
```

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

```
-->match(j:person{name:"john"})-[r1:friend_of]->(n)
       where j.birthday=n.birthday return n.name,n.birthday
```

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

-->

Database Technologies: Neo4j Assignment 3 Complex pattern Queries:

1. Library Database :

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

```
MATCH (a:Reader)-[r:RECOMMENDED]->(b:Book)
WHERE b.title="Tinker Tailor Soldier Spy" or b.title="Our Man in Havana"
RETURN a
```

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

```
match (a:Reader) where not (a:Reader)-[:RECOMMENDED]->(:Book) return a
```

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

```
MATCH (b:Book)-[r:ISSUED_BY]->(a:Reader)
RETURN b.title,COUNT(b)

MATCH (b:Book)-[r:ISSUED_BY]->(a:Reader)
where max(count(b))
RETURN b.title, COUNT(b)
```

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

```
MATCH (a:Reader{name:"Lan"})-[r:RECOMMENDED]->(b:Book)
WHERE count(r)>0
RETURN b

MATCH (a:Reader{name:"Lan" })-[r:RECOMMENDED]->(b:Book)-[rr:ISSUED_BY]-
>(rd:Reader)
RETURN a,r,rd,rr,b
```

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

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

g) List the names of voracious readers in our library
[Voracious means the reader who haven't issued any book]

```
MATCH (a:Reader)
WHERE NOT (:Book)-[:ISSUED_BY]->(a:Reader)
RETURN a.name
```

3. Employee Database:

a) List the name of employees in department" IT".

```
-->match (e:Employee)-[:Works_in]->(:Department{Name:"IT"})
return  e.Name
```

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

```
--->match  (d:Department{Name:'IT'})<-[:Controlled_by]-(p:Projects)<-
[:Assigned_to]-(e:Employee),
(e:Employee)-[:Works_in]->(d:Department)
return d,p,e
```

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

```
--->match (e:Employee{Name:'Harry'})-[:Project_manager]->(p:Projects)-
[:Controlled_by]->(d:Department)
return e,p,d
```

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

```
--->match(e:Employee{Name:'Harry'})--(s:Skillset)--(ee:Employee) return ee,s
```

5. Social Network Database:

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

```
-->match(n:person)-[r4:creates]->(t:timeline) return max(n)
```

b) List all friends of John’s friend, Tom

```
-->match(n:person)<-[r4:friend_of]-(j:person) return n
```

c) List the people with maximum friends

```
-->match (n:person)-[r1:friend_of]->(m:person) return max(n)
```

d) List the people who are part of more than 3 groups.

```
-->match(m:person)-[r3:belongs_to]->(g:group) with m, count(*)  as cnt
where cnt>3 return m.name
```