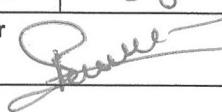


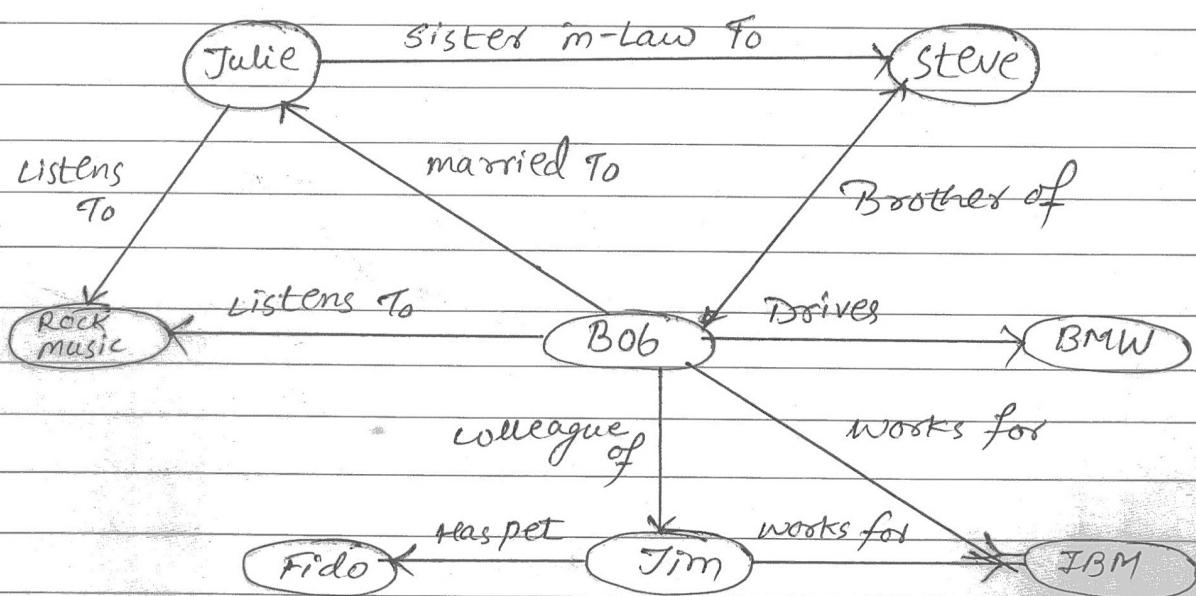
		Hope Foundation's Finolex Academy of Management and Technology, Ratnagiri Information Technology Department		
Subject name: Big Data Analytics			Subject Code: ITL801	
Class	BE IT	Semester – VIII (CBGS)	Academic year: 2019-20	
Name of Student	Kazi Jawwad A Rahim		QUIZ Score :	
Roll No	28	Assignment/Experiment No.	03	
Title: Creating graph database on Neo4j and displaying the relationships of graph.				
1. Course objectives applicable COB3.Understand various concepts of finding similar items and mining data streams COB4.Study Page Rank in Link Analysis and concepts of Handling larger datasets COB5.Study various clustering algorithms and Recommendation systems.				
2. Course outcomes applicable: CO3 -Interpret business models and scientific computing paradigms, and apply software tools for big data analytics CO4-Implement use of combiners to consolidate results and ability to handle larger datasets.				
3. Learning Objectives: <ol style="list-style-type: none"> 1. Understand the Neo4j Database application domain 2. Install and run Neo4j on the machine 3. Create graph database and display the relations of graph 4. Analyze the queries for creating relationship between nodes 				
4. Practical applications of the assignment/experiment: Neo4J is a graph Database, used in relationship based analysis on social networks				
5. Prerequisites: <ol style="list-style-type: none"> 1. Knowledge of SQL 2. Knowledge of Graph Databases 				
6. Hardware Requirements: <ol style="list-style-type: none"> 1. PC with 4GB RAM, 500GB HDD 				
7. Software Requirements: <ol style="list-style-type: none"> 1. Access to Neo4j Sandbox 2. Google chrome version 6.23 or higher 				
8. Quiz Questions (if any): (Online Exam will be taken separately batchwise, attach the certificate/ Marks obtained) <ol style="list-style-type: none"> 1. What is a Graph Database? 2. Which Query language does graph database use? 3. Can you install Neo4J on windows? If yes, explain the process. 4. What relations were plotted on the console ? 				
9. Experiment/Assignment Evaluation:				
Sr. No.	Parameters		Marks obtained	Out of
1	Technical Understanding (Assessment may be done based on Q & A <u>or</u> any other relevant method.) Teacher should mention the other method used -		05	6
2	Neatness/presentation		01	2
3	Punctuality		02	2
Date of performance (DOP)	22-01-2020	Total marks obtained	08	10
Date of checking (DOC)	07-02-2020	Signature of teacher		



Theory :

A graph database, also called a graph-oriented database, is a type of NoSQL database that uses graph theory to store, map and query relationships.

A graph database is essentially a collection of nodes and edges. Each node represents an entity (such as a person or business) and each edge represents a connection or relationship between two nodes. Every node in a graph database is defined by a unique identifier, a set of outgoing edges and/or incoming edges and a set of properties expressed as key/value pairs. Each edge is defined by a unique identifier, a starting place and/or ending-place node and a set of properties. The mantra of graph database enthusiasts is "if you can whiteboard it, you can graph it."





Graph databases are well-suited for analyzing interconnections, which is why there has been a lot of interest in using graph databases to mine data from social media. Graph databases are also useful for working with data in business disciplines that involve complex relationships and dynamic schema, such as supply chain management, identifying the source of an IP telephony issue and creating "customers who bought this also looked at..." recommendations.

The concept behind graphing a database is often credited to 18th-century mathematician Leonhard Euler.

2. Precautions :

1. Register on Neo4J first,
2. Request for creating sandbox, it will take time

3. Installation Steps / Performance Steps -

- 1) Sign in to Neo4j Sandbox installer.
- 2) Run the cypher queries
- 3) View the database relationships:
- 4) NoSQL Graph database Neo4j (The Movie graph)

The screenshot shows the Neo4j browser interface. On the left, there is a code editor containing Cypher query code. On the right, the results of the query are displayed as a graph visualization titled "The Movie Graph".

Code (Cypher):

```
1 CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})
2 CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
3 CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
4 CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
5 CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
6 CREATE (AndyW:Person {name:'Andy Wachowski', born:1967})
7 CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
```

:play movie graph

The Movie Graph

Create

To the right is a giant code block containing a single Cypher query statement composed of multiple CREATE clauses. This will create the movie graph.

1. Click on the code block

Result (Graph):

```
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 (AndyW:Person {name:'Andy Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
```

4.Observations

1)

Find the actor named "Tom Hanks"...

```
① MATCH (tom {name: "Tom Hanks"}) RETURN tom
```

Find the movie with title "Cloud Atlas"...

```
② MATCH (cloudAtlas {title: "Cloud Atlas"}) RETURN cloudAtlas
```

Find 10 people...

```
③ MATCH (people:Person) RETURN people.name LIMIT 10
```

Find movies released in the 1990s...

```
④ MATCH (nineties:Movie) WHERE nineties.released >= 1990 AND nineties.released < 2000 RETURN nineties.title
```

2)

List all Tom Hanks movies...

```
④ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(tomHanksMovies) RETURN tom, tomHanksMovies
```

Who directed "Cloud Atlas"?

```
④ MATCH (cloudAtlas {title: "Cloud Atlas"})-<[:DIRECTED]->(directors) RETURN directors.name
```

Tom Hanks' co-actors...

```
④ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m)-<[:ACTED_IN]->(coActors) RETURN coActors.name
```

How people are related to "Cloud Atlas"...

```
④ MATCH (people:Person)-[relatedTo]->(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
```

3)

Movies and actors up to 4 "hops" away from Kevin Bacon

```
④ MATCH (bacon:Person {name: "Kevin Bacon"})-[*1..4]-(hollywood)  
RETURN DISTINCT hollywood
```

Bacon path, the shortest path of any relationships to Meg Ryan

```
④ MATCH p=shortestPath(  
  (bacon:Person {name: "Kevin Bacon"})-[*]-(meg:Person {name: "Meg Ryan"})  
)  
RETURN p
```

Note you only need to compare property values like this when first creating relationships

4)

Extend Tom Hanks co-actors, to find co-co-actors who haven't worked with Tom Hanks...

```
④ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m)-<[:ACTED_IN]->(coActors),  
       (coActors)-[:ACTED_IN]->(m2)-<[:ACTED_IN]->(cocoActors)  
WHERE NOT (tom)-[:ACTED_IN]->()-<[:ACTED_IN]->(cocoActors) AND tom <> cocoActors  
RETURN cocoActors.name AS Recommended, count(*) AS Strength ORDER BY Strength DESC
```

Find someone to introduce Tom Hanks to Tom Cruise

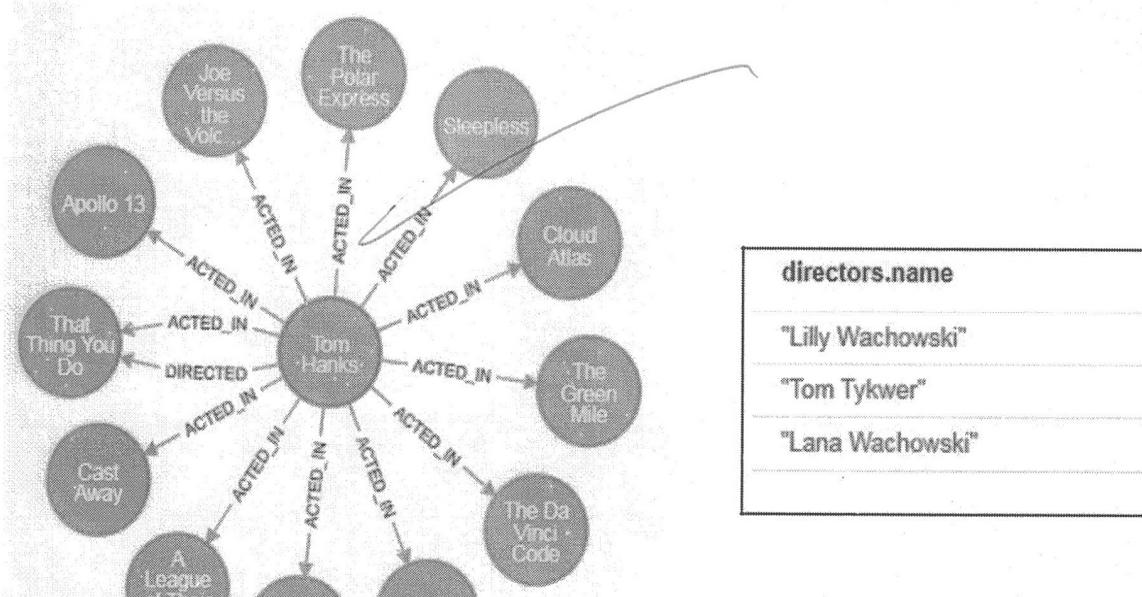
```
④ MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m)-<[:ACTED_IN]->(coActors),  
       (coActors)-[:ACTED_IN]->(m2)-<[:ACTED_IN]->(cruise:Person {name: "Tom Cruise"})  
RETURN tom, m, coActors, m2, cruise
```

5. Results:

Output of SET 1:

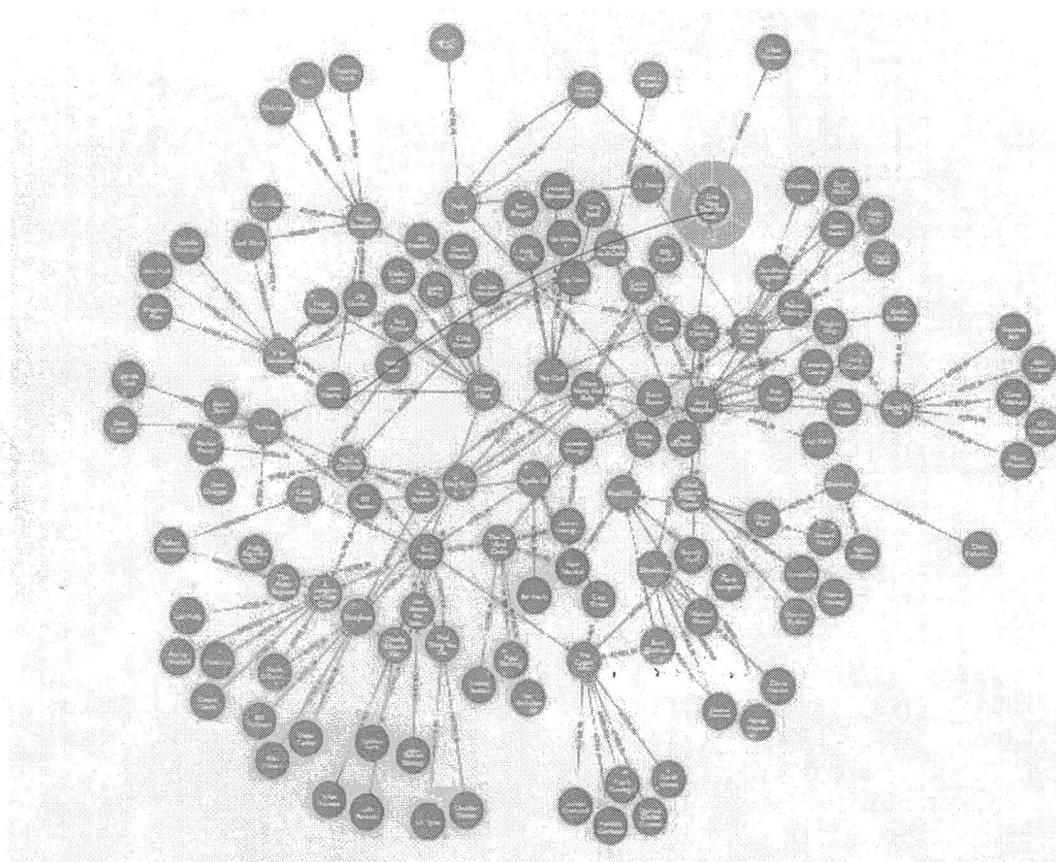
people.name	nineties.title
"Keanu Reeves"	"The Matrix"
"Keanu Reeves"	"The Matrix"
"Carrie-Anne Moss"	"The Devil's Advocate"
"Laurence Fishburne"	"A Few Good Men"
"Hugo Weaving"	"As Good as It Gets"
"Lilly Wachowski"	"What Dreams May Come"
"Lana Wachowski"	"Snow Falling on Cedars"
"Joel Silver"	"You've Got Mail"
"Emil Eifrem"	"Sleepless in Seattle"
"Charlize Theron"	

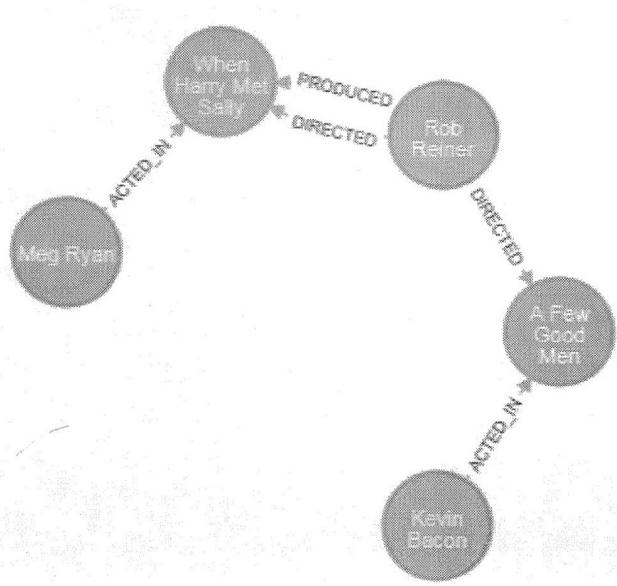
Output of SET 2:



coActors.name	people.name	Type[relatedTo]	relatedTo
"Sam Rockwell"	"Lilly Wachowski"	"DIRECTED"	{ }
"David Morse"	"Hugo Weaving"	"ACTED_IN"	{ "roles": ["Bill Smoke", "Haskell Moore", "Tadeusz Kesselring", "Nurse Naoko", "Boardman Nephil", "Old Georgie"] } }
"Patricia Clarkson"			
"James Cromwell"			
"Michael Clarke Duncan"			
"Gary Sinise"			
"Bonnie Hunt"			
"Dave Chappelle"			
"Meg Ryan"			
"Greg Kinnear"			
"Steve Zahn"			
"Parker Posey"			
"Paul Bettany"			

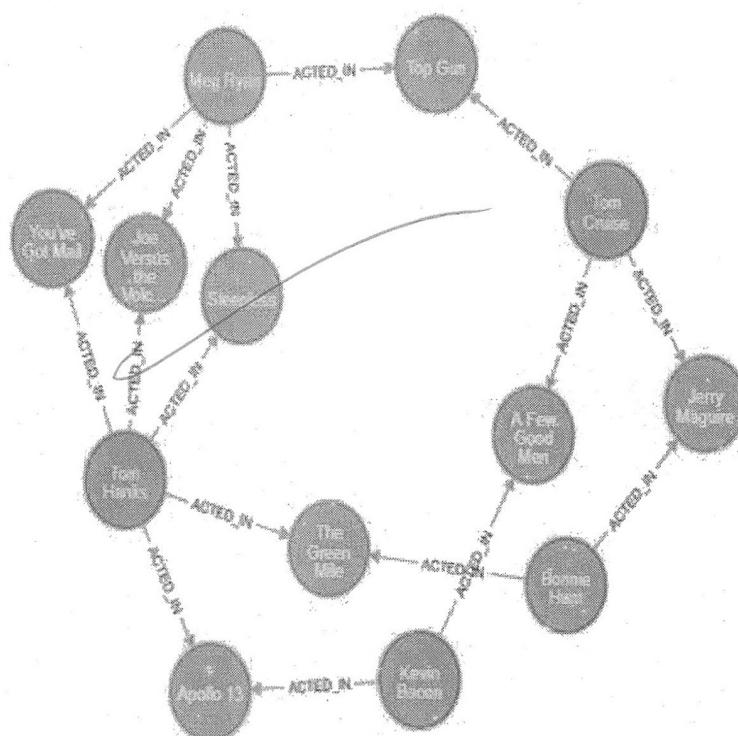
Output of SET 3:





Output of SET 4:

Recommended	Strength
"Tom Cruise"	5
"Zach Grenier"	5
"Cuba Gooding Jr."	4
"Keanu Reeves"	4
"Carrie Fisher"	3
"Billy Crystal"	3
"Bruno Kirby"	3
"Val Kimer"	3
"Kelly McGillis"	3
"Tom Skerrit"	3
"Anthony Edwards"	3
"Jack Nicholson"	3
"Laurence Fishburne"	3
"Carrie-Anne Moss"	3
"Oliver Platt"	2
"Michael Sheen"	2
"Frank Langella"	2
"Max von Sydow"	1
"Rick Yune"	1



8. References :

- [1] <https://neo4j.com/developer/graph-database/>
- [2] <http://whatis.techtarget.com/definition/graph-database>



Learning Outcomes Achieved:

1. Neo4j Database was created.
2. Nodes with relations are created.
3. The graph for node and their relation is plotted.
4. The queries for adding node, adding relationships are analysed.
5. Use of Neo4j database for relations and graph type data is studied here.

Conclusion:

1. Applications of the studied technique in industry
 - a. Neo4j is prominent Graph Database.
 - b. For social networking, online shopping sites use graph relationships.
2. Engineering Relevance
 - a. Social networks have more applications of Neo4j database.
3. Skills Developed
 - a. Creating a Neo4j Sandbox on Neo4j cloud.
 - b. Creating a graph database, creating records and displaying relationships.