

Applied Databases

Topic 6 Exercise Sheet

Part 1

- Run Neo4j as follows:
 - Open a Windows Command prompt/PowerShell and change to the [bin](#) folder of the Neo4j installation.
 - Run [neo4j console](#)

```
C:\>cd \Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\bin

C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\bin>neo4j console
Directories in use:
home:           C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3
config:         C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\conf
logs:           C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\logs
plugins:        C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\plugins
import:         C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\import
data:           C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\data
certificates:   C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\certificates
licenses:       C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\licenses
run:            C:\Users\GHarrison\Documents\neo4j-community-4.3.3-windows\neo4j-community-4.3.3\run
Starting Neo4j.
2021-09-29 19:21:36.589+0000 INFO   Starting...
2021-09-29 19:21:38.479+0000 INFO   ===== Neo4j 4.3.3 =====
2021-09-29 19:21:39.424+0000 INFO   Performing postInitialization step for component 'security-users' with version 3 and
status CURRENT
2021-09-29 19:21:39.425+0000 INFO   Updating the initial password in component 'security-users'
2021-09-29 19:21:40.085+0000 INFO   Bolt enabled on 127.0.0.1:7687.
2021-09-29 19:21:40.621+0000 INFO   Remote interface available at http://localhost:7474/
2021-09-29 19:21:40.623+0000 INFO   Started.
```

- Then open a browser to <http://localhost:7474>

- Create the following nodes with a label :Student with the following properties:

- o name: "Tom"
- o sid: "G001"
- o age: 23
- o sex: "M"
- o address: "Galway"
- o hair: "brown"
- o email: "tom@gmail.com"

```
create(:Student{
sid:"G001",name:"Tom",age:23,sex:"M",address:"Galway",hair:"brown",email:"tom@gmail.com"})
```

- o name: "Sean"
- o sid: "G002"
- o age: 19
- o sex: "M"
- o address: "Galway"
- o email: "sean@gmail.com"

```
create(:Student{
sid:"G002",name:"Sean",age:19,sex:"M",address:"Galway",email:"sean@gmail.com"})
```

- o name: "Bob"
- o sid: "G003"
- o age: 22
- o sex: "M"
- o address: "Mayo"
- o email: "bob123@hotmail.com"
- o twitter: "@bob123"

```
create(:Student{
sid:"G003",name:"Bob",age:22,sex:"M",address:"Mayo",email:"bob123@gmail.com",twitter:"@bob123"})
```

- o name: "Mary"
- o sid: "G004"
- o age: 20
- o sex: "F"
- o address: "Mayo"
- o hair: "blonde"
- o email: "mary19@gmail.com"
- o twitter: "@mary19"
- o snapchat: "mary19"

```
create(:Student{  
sid:"G004",name:"Mary",age:20,sex:"F",address:"Mayo",hair  
:"blonde",email:"mary19@gmail.com",twitter:"@mary19",snap  
chat:"mary19"})
```

- o name: "Alice"
- o sid: "G005"
- o age: 28
- o sex: "F"
- o address: "Roscommon"
- o email: "alice@hotmail.com"
- o snapchat: "alice123"

```
create(:Student{sid:"G005",name:"Alice",age:28,sex:"F",ad  
dress:"Roscommon",email:"alice@hotmail.com",snapchat:"ali  
ce123"})
```

- o name: "Pat"
- o sid: "G006"
- o age: 24
- o sex: "M"
- o address: "Roscommon"
- o email: "pat@hotmail.com"
- o twitter: "patABC"

```
create(:Student{  
sid:"G006",name:"Pat",age:24,sex:"M",address:"Roscommon",  
email:"pat@hotmail.com",snapchat:"patABC"})
```

- Create the following nodes with a label :Lecturer with the following properties:

- name: "Alan"
- sid: "L001"
- age: 57
- sex: "M"
- address: "Galway"
- email: "alan@gmit.ie",
- twitter: "@alan"

```
create(:Lecturer{
sid:"L001",name:"Alan",age:57,sex:"M",address:"Galway",email
:"alan@gmit.ie"})
```

- name: "Mary"
- sid: "L002"
- age: 47
- sex: "F"
- address: "Mayo"
- email: "mary@gmit.ie"
- hair: "brown"

```
create(:Lecturer{
sid:"L002",name:"Mary",age:47,sex:"F",address:"Mayo",email:"ma
ry@gmit.ie"})
```

- Find the average age of Students, rounded to the nearest whole number.

```
MATCH (n:Student) RETURN round(avg(n.age))
```

- Show the name of each student and his/her age.

```
MATCH (n:Student) RETURN n.name,n.age
```

- Find the age of the youngest Student.

```
MATCH (n:Student) RETURN min(n.age)
```

- Show the names of students who have a *twitter* attribute.

```
MATCH (n:Student) WHERE n.twitter IS NOT NULL
RETURN n.name
```

- Show the number of students who have a *twitter* attribute.
`MATCH(n:Student) RETURN count(n.twitter)`
- Show the average of age of people in their 20s, 30s and 40s rounded to one decimal place.
`MATCH(n) WHERE n.age>=20 AND n.age<=49
RETURN round(avg(n.age),1)`
- Show all the properties for the Student *Tom*.
`MATCH (n:Student{name:"Tom"}) return keys(n)`
- Increase everyone's age by 1.
`match(n) set n.age=n.age+1 return n`
- Return the name and age of all males living in Galway.
`MATCH (n) where n.sex="M" and n.address="Galway" RETURN
n.name, n.age`

- Create the following nodes with both :Student and :Lecturer labels

- o name: "Yvonne"
 - age: 37
 - sex: "F"
 - address: "Galway"
 - email: yvonne@gmit.ie
 - twitter: "@yv12"
- o name: "Walter"
 - age: 44
 - address: "Galway"
 - email: walter@gmit.ie
 - hair: "black"

```
create (:Lecturer:Student{name:"Yvonne",age:37,sex:"F",address:
"Galway",email:"yvonne@gmit.ie",twitter:"yv12"})
```

```
create (:Lecturer:Student{name:"Walter",age:44,sex:"M",address:
"Galway",email:"walter@gmit.ie",hair:"black"})
```

or

```
CREATE (:Student:Lecturer{name:"Yvonne",age:37,sex:"F",address:
"Galway",email:"yvonne@gmit.ie",twitter:"@yv12"}), (:Student:Le
cturer{name:"Walter",age:44,address:"Galway",email:"walter@gmi
t.ie",hair:"black"})
```

- Show the name, age and hair colour of everyone who is both a Student and a Lecturer.

```
match(n:Student:Lecturer) return n.name, n.age, n.hair
```

or

```
MATCH(p) WHERE p:Student AND p:Lecturer  
RETURN p.name, p.hair
```

- Update the *twitter* attribute of all lectures to have GMIT after their existing twitter name.
E.g. “@alan” should become “@alanGMIT”.

```
match(n:Lecturer) set n.twitter = n.twitter+"@GMIT"
return n
```

Only updates the *twitter* attribute to have “@GMIT” appended for Lecturer nodes, if they currently have a *twitter* attribute.

- Find the average age of Males and find the youngest Male(s).
Then return the name (as *Name*) and age (as *Age*) of the youngest Male(s) as well as the average age of Males (as *AverageAge*) and the difference in age between the youngest Male(s) and the average age (as *Difference*).

E.g., If the average age of Males was 30, and the youngest Male was called “Tony” aged 20, the following should be returned:

Name	Age	AverageAge	Difference
Tony	20	30	10

```
MATCH(p{sex:"M"}) WITH avg(p.age) AS avgAge
MATCH(p1{sex:"M"}) WITH min(p1.age) AS minAge, avgAge
MATCH(p2{sex:"M"}) WHERE p2.age = minAge
RETURN p2.name, p2.age, avgAge, avgAge-
p2.age AS Difference
```


Part 2

- Use a new database.
- In the Neo4j Browser, create a new (blank) database and type the following command:

```
tdb$ :play movies
```

- This will return the following:

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
2. Notice it gets copied to the editor above
3. Click the editor's play button to execute
4. Wait for the query to finish

WARNING: This adds data to the current database, each time it is run!

2/8 < >

- Go to page 2 and follow steps 1 to 4.
- A series of 171 nodes (representing Movies and People) and 253 relationships (such as ACTED_IN, DIRECTED etc.) between the nodes should now be created.
- Type MATCH(n) RETURN n to see all nodes and relationships:

Database Information

Use database: tdb

Node Labels: tdb\$ Movie, Person

Relationship Types: tdb\$ ACTED_IN, DIRECTED, FOLLOWED, PRODUCED, REVIEWED, WROTE

Property Keys: born, name, rating, released, roles, summary, tagline, title

Connected as: Username: neo4j, Roles: -

tdb\$ MATCH(n) RETURN n

Displaying 171 nodes, 253 relationships.

- Show each movie node for movies that were released between 2000 and 2010

```
match(m:Movie) where m.released >= 2000 and m.released <= 2010
return m
```

- Set an attribute called olderThan70 to true for all Persons born in the 1930s.

```
match(p:Person) where p.born >= 1930 and p.born <= 1939
set p.olderThan70 = true
return p
```

- Show the movie title and the year it was released for the first 10 movies in alphabetical order.

```
match(m:Movie) RETURN m.title, m.released ORDER BY m.title LIMIT 10
```

- Show the unique years in which movies were released in chronological order.

```
match(m:Movie) RETURN distinct m.released ORDER BY m.released
```

- Show the title and tagline for movies released in 1999.

```
match(m:Movie{released:1999}) return m.title, m.tagline
```

or

```
match(m:Movie) WHERE m.released=1999 return m.title, m.released
```

- Show the names of the people (as *People*) and the year they were born (as *YOB*) for everyone older than "Robin Williams".

```
MATCH(p:Person{name:"Robin Williams"}) with p.born as williamsBorn
MATCH(p1:Person) WHERE p1.born < williamsBorn
RETURN p1.name AS Person, p1.born as YOB
ORDER BY YOB, Person
```

- Show the number of movies released in 2006 (as *Releases_in_2006*).

```
MATCH(m:Movie{released:2006}) return count(m) AS Releases_in_2006
```

- Show the name (as *Name*) and born (as *YOB*) the youngest Person(s).

```
MATCH(p:Person) WITH max(p.born) as youngest
MATCH(p1:Person) WHERE p1.born = youngest
RETURN p1.name as Name, p1.born as YOB
```