NTU Database Management System from SQL to NoSQL
Homework 6

R10725059 資管碩二 謝佳穎

Part I: MongoDB

Task1

sudo mongod --dbpath=~/mongodb-data --port=27017
mongosh
show dbs
use hw6

```
Current Mongosh Log ID: 648ac3041703fd94a9dccc6f
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&app
Name=mongosh+1.9.1
Using MongoDB:          6.0.6
Using Mongosh:          1.9.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-06-15T15:50:29.469+08:00: Access control is not enabled for the database. Read and write access to
 data and configuration is unrestricted
   2023-06-15T15:50:29.469+08:00: You are running this process as the root user, which is not recommended
   2023-06-15T15:50:29.469+08:00: This server is bound to localhost. Remote systems will be unable to conn
ect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serv
e responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the
server with --bind_ip 127.0.0.1 to disable this warning
------

test> show dbs
admin          40.00 KiB
config         84.00 KiB
local          72.00 KiB
task-manager  112.00 KiB
test> use hw6
switched to db hw6
hw6> show dbs
admin          40.00 KiB
config         84.00 KiB
local          72.00 KiB
task-manager  112.00 KiB
hw6> db.createCollection("students")
{ ok: 1 }
```

cd 2023-DBMS
mongoimport --db=hw6 --collection=students --type=csv --headerline
--file=./hw6/DBMS_student_list.csv

```
~/Desktop/2023-DBMS  mongoimport --db=hw6 --collection=students --type=csv --headerline --file=./hw6//DB
MS_student_list.csv
2023-06-15T15:54:47.234+0800    connected to: mongodb://localhost/
2023-06-15T15:54:47.257+0800    98 document(s) imported successfully. 0 document(s) failed to import.
```

Task2

db.students.find({ 身份:{$nin: ["旁聽生"]}, 系所: "資管系", 年級: {$nin: [3,4]}, 姓名: {$nin: ["劉正宇 (LIOU, CHENG-YU)"]} })

```
hw6> db.students.find( { 身份:{$nin: ["旁聽生"]}, 系所: "資管系", 年級: {$nin: [3,4]}, 姓名: {$nin: ["劉正宇 (LIOU, CHENG-YU)"]} } )
[
  {
    _id: ObjectId("648ac3c7ba701dc0dd854621"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 1,
    '學號': 'R11725044',
    '姓名': '鄧鈺儒 (TENG, YU-JU)',
    '信箱': 'r11725044@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)'
  },
  {
    _id: ObjectId("648ac3c7ba701dc0dd854622"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 1,
    '學號': 'R11725047',
    '姓名': '林品歷 (LIN, PIN-LI)',
    '信箱': 'r11725047@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)'
  },
  {
    _id: ObjectId("648ac3c7ba701dc0dd854623"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 1,
    '學號': 'R11725058',
    '姓名': '吳品萱 (PIN-HSUAN WU)',
    '信箱': 'r11725058@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)'
  },
  {
    _id: ObjectId("648ac3c7ba701dc0dd854625"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 2,
    '學號': 'R10725059',
    '姓名': '謝佳穎 (HSIEH, CHIA-YING)',
    '信箱': 'r10725059@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)'
  }
]
```

Task3

db.students.aggregate([{$group:{_id: {department: "$系所", grade: "$年級"}, total:{$sum:1}}}])

```
hw6> db.students.aggregate( [{$group:{_id: {department: "$系所", grade: "$年級"}, total:{$sum:1}}}] )
[
  { _id: { department: '統計碩士學程', grade: 1 }, total: 2 },
  { _id: { department: '土木系電輔組', grade: 1 }, total: 3 },
  { _id: { department: '資管系', grade: 1 }, total: 4 },
  { _id: { department: '農經系', grade: 2 }, total: 1 },
  { _id: { department: '心理系', grade: 4 }, total: 2 },
  { _id: { department: '資管系', grade: 2 }, total: 3 },
  { _id: { department: '資料科學學程', grade: 2 }, total: 2 },
  { _id: { department: '生物機電系', grade: 4 }, total: 1 },
  { _id: { department: '電信所', grade: 1 }, total: 3 },
  { _id: { department: '統計碩士學程', grade: 2 }, total: 2 },
  { _id: { department: '生機系', grade: 1 }, total: 1 },
  { _id: { department: '國企系', grade: 2 }, total: 1 },
  { _id: { department: '土木系水利組', grade: 1 }, total: 2 },
  { _id: { department: '生機系', grade: 2 }, total: 2 },
  { _id: { department: '工科海洋系', grade: 3 }, total: 1 },
  { _id: { department: '生傳系', grade: 4 }, total: 1 },
  { _id: { department: '電機系', grade: 3 }, total: 4 },
  { _id: { department: '資工系', grade: 3 }, total: 6 },
  { _id: { department: '電信所', grade: 2 }, total: 3 },
  { _id: { department: '電機系', grade: 6 }, total: 1 }
]
Type "it" for more
hw6> it
[
  { _id: { department: '應數所', grade: 1 }, total: 1 },
  { _id: { department: '基蛋所', grade: 3 }, total: 1 },
  { _id: { department: '資料科學學程', grade: 1 }, total: 1 },
  { _id: { department: '基蛋所', grade: 2 }, total: 1 },
  { _id: { department: '心理系一般組', grade: 2 }, total: 1 },
  { _id: { department: '資工系', grade: 2 }, total: 5 },
  { _id: { department: '電機系', grade: 2 }, total: 2 },
  { _id: { department: '生醫電資所', grade: 1 }, total: 4 },
  { _id: { department: '資工系', grade: 1 }, total: 2 },
  { _id: { department: '醫工系', grade: 4 }, total: 1 },
  { _id: { department: '資料科學博士', grade: 1 }, total: 1 },
  { _id: { department: '經濟系', grade: 4 }, total: 4 },
  { _id: { department: '物理系', grade: 4 }, total: 2 },
  { _id: { department: '資管系', grade: 3 }, total: 1 },
  { _id: { department: '電機系', grade: 1 }, total: 16 },
  { _id: { department: '光電所', grade: 2 }, total: 1 },
  { _id: { department: '網媒所', grade: 1 }, total: 1 },
  { _id: { department: '機械系', grade: 4 }, total: 1 },
  { _id: { department: '土木系水利組', grade: 2 }, total: 1 },
  { _id: { department: '系統生物學程', grade: 3 }, total: 1 }
]
Type "it" for more
hw6> it
[
  { _id: { department: '機械系系控組', grade: 1 }, total: 1 },
  { _id: { department: '公衛系', grade: 3 }, total: 2 },
  { _id: { department: '電機資安碩班', grade: 2 }, total: 1 },
  { _id: { department: '資工系', grade: 4 }, total: 1 }
]
```

Task4

db.students.updateMany({}, {$set:{join_date: "2023-03-01"}})
db.students.find({ 身份:{$nin: ["旁聽生"]}, 系所: "資管系", 年級: {$nin: [3,4]}, 姓名: {$nin: ["劉正宇 (LIOU, CHENG-YU)"]} })

```
hw6> db.students.updateMany({}, {$set:{join_date: "2023-03-01"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 98,
  modifiedCount: 98,
  upsertedCount: 0
}
```

```
hw6> db.students.find( { 身份:{$nin: ["旁聽生"]}, 系所: "資管系", 年級: {$nin: [3,4]}, 姓名: {$nin: ["劉正宇 (LIOU, CHENG-YU)"]} } )
[
  {
    _id: ObjectId("648ac3c7ba701dc0dd854621"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 1,
    '學號': 'R11725044',
    '姓名': '鄧鈺儒 (TENG, YU-JU)',
    '信箱': 'r11725044@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)',
    join_date: '2023-03-01'
  },
  {
    _id: ObjectId("648ac3c7ba701dc0dd854622"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 1,
    '學號': 'R11725047',
    '姓名': '林品歷 (LIN, PIN-LI)',
    '信箱': 'r11725047@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)',
    join_date: '2023-03-01'
  },
  {
    _id: ObjectId("648ac3c7ba701dc0dd854623"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 1,
    '學號': 'R11725058',
    '姓名': '吳品萱 (PIN-HSUAN WU)',
    '信箱': 'r11725058@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)',
    join_date: '2023-03-01'
  },
  {
    _id: ObjectId("648ac3c7ba701dc0dd854625"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 2,
    '學號': 'R10725059',
    '姓名': '謝佳穎 (HSIEH, CHIA-YING)',
    '信箱': 'r10725059@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)',
    join_date: '2023-03-01'
  }
]
```

Task5

mongoimport --db=hw6 --collection=students --type=csv --headerline --file=./hw6/new_student_list.csv --mode insert

```
~/Desktop/2023-DBMS  mongoimport --db=hw6 --collection=students --type=csv --headerline --file=./hw6/new_student_list.csv --mode insert

2023-06-15T16:01:04.371+0800    connected to: mongodb://localhost/
2023-06-15T16:01:04.393+0800    3 document(s) imported successfully. 0 document(s) failed to import.
```

```
hw6> db.students.countDocuments()
101
hw6> db.students.find({姓名: {$in:["謝佳穎 (HSIEH, CHIA-YING)", "小紅", "小黃", "小綠"]}})
[
  {
    _id: ObjectId("648ac3c7ba701dc0dd854625"),
    '身份': '學生',
    '系所': '資管系',
    '年級': 2,
    '學號': 'R10725059',
    '姓名': '謝佳穎 (HSIEH, CHIA-YING)',
    '信箱': 'r10725059@ntu.edu.tw',
    '班別': '資料庫系統-從SQL到NoSQL (EE5178)',
    join_date: '2023-03-01'
  },
  {
    _id: ObjectId("648ac540faa82cbc1e4ffb45"),
    '身份': '旁聽生',
    '系所': '電機所',
    '年級': 2,
    '學號': 'R10123456',
    '姓名': '小紅',
    join_date: '2023-06-02'
  },
  {
    _id: ObjectId("648ac540faa82cbc1e4ffb46"),
    '身份': '觀察者',
    '系所': '電信所',
    '年級': 1,
    '學號': 'R11123001',
    '姓名': '小綠',
    join_date: '2023-06-02'
  },
  {
    _id: ObjectId("648ac540faa82cbc1e4ffb47"),
    '身份': '學生',
    '系所': '物理系',
    '年級': 3,
    '學號': 'B09987653',
    '姓名': '小黃',
    join_date: '2023-06-02'
  }
]
```

Task6

```
calsum= function(thedate){
   db.students.aggregate([
      {$match: {join_date: thedate}},
      {$group: {
         _id:{ department: "$系所", grade: "$年級" },
         total: { $sum: 1 }
      }},
      {$group: {
         "_id":"tally",
         groupby_result: { $push: "$$ROOT" }
      }},
      {$merge:{
         into:"students",
```

```
            whenMatched: "replace",
            whenNotMatched: "insert"
        }}
    ])
}
```

- 依據指定日期進行系級分類計算人數之依據
- 依據系所、年級進行group by後，計算分組人數
- 之後將結果object之id命名為tally, 結果object push至另一個array名為groupby_result
- 最後將此document併至students，並設定replace以讓之後計算的結果可以直接覆蓋，達成incremental update。

```
calsum("2023-03-01")
db.students.find({"_id":"tally"})
calsum("2023-06-02")
db.students.find({"_id":"tally"})
```

```
hw6> calsum= function(thedate){
...     db.students.aggregate([
...         {$match: {join_date: thedate}},
...         {$group: {
...             _id:{ department: "$系所", grade: "$年級" },
...             total_count: { $sum: 1 }
...         }},
...         {$group: {
...             "_id":"tally",
...             groupby_result: { $push: "$$ROOT" }
...         }},
...         {$merge:{
...             into:"students",
...             whenMatched: "replace",
...             whenNotMatched: "insert"
...     }}
...     ])
...         }}
...     ])
... }
[Function: calsum]
hw6> calsum("2023-03-01")
```

```
_id: "tally"
groupby_result: Array
  0: Object
    _id: Object
        department: "統計碩士學程"
        grade: 1
      total_count: 2
  1: Object
  2: Object
  3: Object
  4: Object
  5: Object
  6: Object
  7: Object
  8: Object
  9: Object
  10: Object
  11: Object
  12: Object
  13: Object
  14: Object
  15: Object
  16: Object
  17: Object
  18: Object
  19: Object
```

```
_id: "tally"
groupby_result: Array
  0: Object
    _id: Object
        department: "物理系"
        grade: 3
      total_count: 1
  1: Object
  2: Object
```

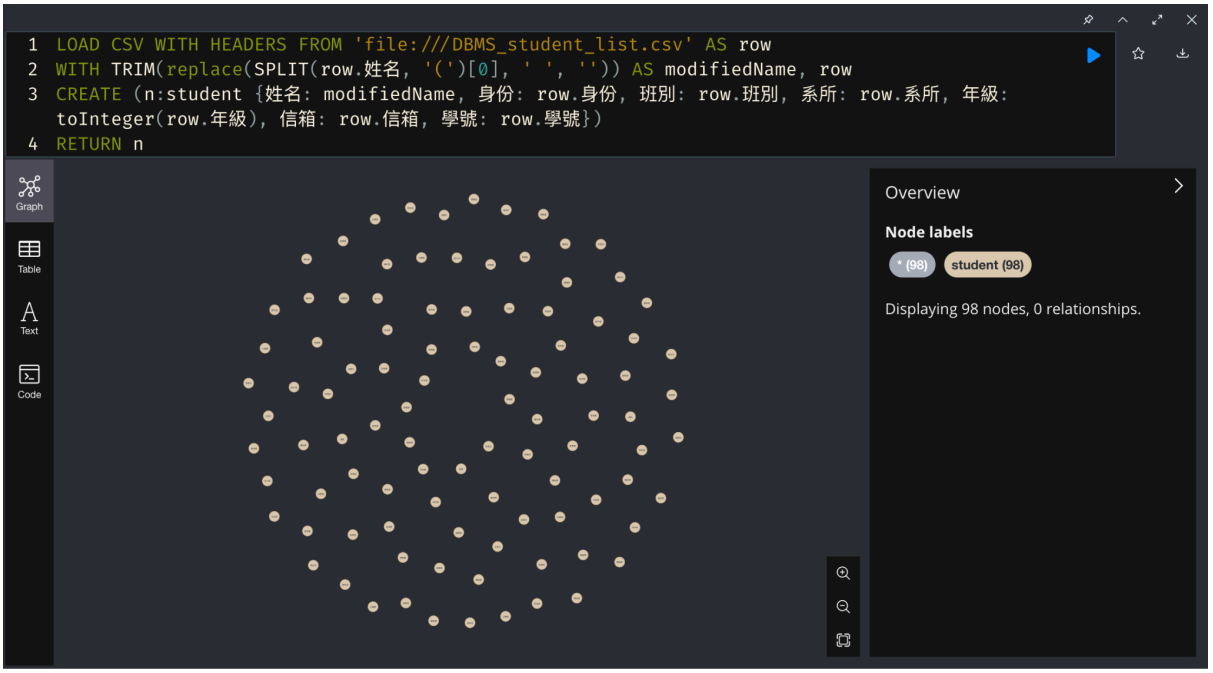

##

Part II: GraphDB Basic

Task1

LOAD CSV WITH HEADERS FROM 'file:///DBMS_student_list.csv' AS row
WITH TRIM(replace(SPLIT(row.姓名, '(')[0], ' ', '')) AS modifiedName, row
CREATE (:student {姓名: modifiedName, 身份: row.身份, 班別: row.班別, 系所: row.系所, 年級: toInteger(row.年級), 信箱: row.信箱, 學號: row.學號})

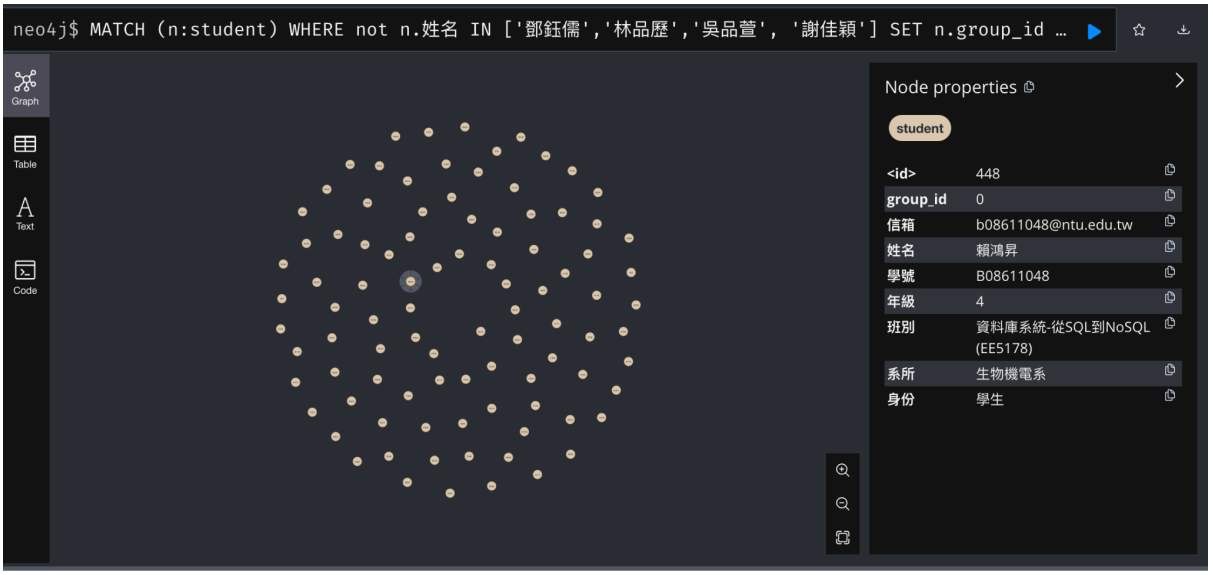
- 將DBMS_student_list.csv 複製到 import資料夾
- 執行上述指令，載入資料入neo4j DB
- 將學生姓名括號與空白部分去除之後再儲存



Task2

MATCH (n:student)
WHERE n.姓名 IN ['鄧鈺儒','林品歷','吳品萱', '謝佳穎']
SET n.group_id = 12
return n

MATCH (n:student)
WHERE not n.姓名 IN ['鄧鈺儒','林品歷','吳品萱', '謝佳穎']
SET n.group_id = 0
return n



Task3

MATCH (n)
WHERE n.group_id = 12
RETURN COLLECT(n) AS propertyList

##

Part III: GraphDB Advanced

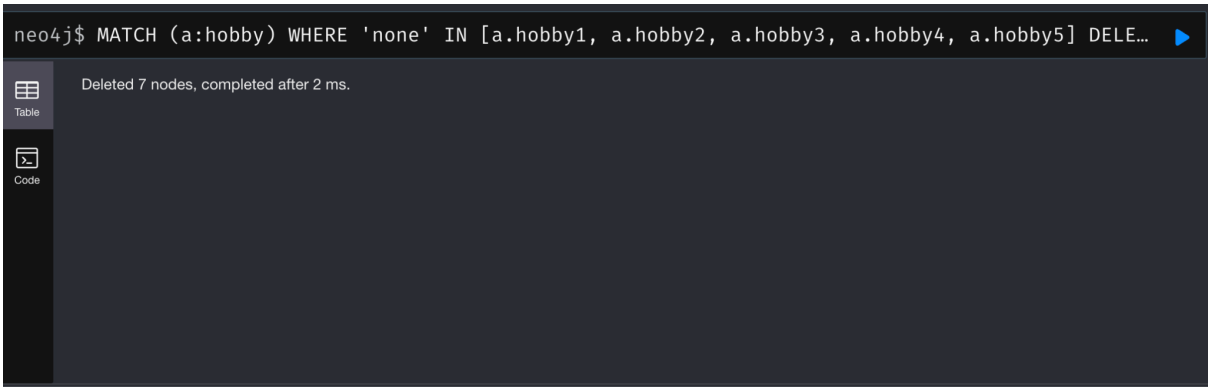
Task1

LOAD CSV WITH HEADERS FROM 'file:///hw6_hobbies.csv' AS row
WITH TRIM(replace(SPLIT(row.姓名, '(')[0], ' ', '')) AS modifiedName, row
CREATE (:hobby {姓名: modifiedName, 學號: row.學號, hobby1: row.hobby1, hobby2: row.hobby2, hobby3: row.hobby3, hobby4: row.hobby4, hobby5: row.hobby5})

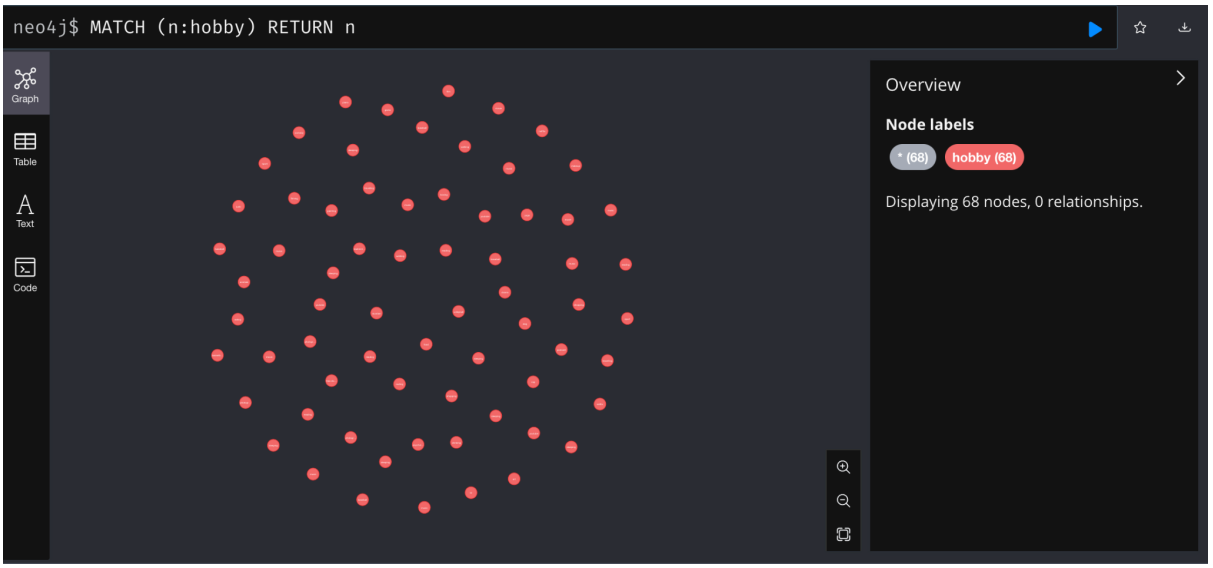
- 將hw6_hobbies.csv 複製到 import資料夾
- 執行上述指令，載入資料入neo4j DB
- 將學生姓名括號與空白部分去除之後再儲存

```
MATCH (a:hobby)
WHERE a.hobby1 = 'none' OR a.hobby2 = 'none' OR a.hobby3 = 'none' OR
a.hobby4 = 'none' OR a.hobby5 = 'none'
delete a
```

- 移除興趣為none的學生




```
MATCH (n:hobby)
RETURN n
```

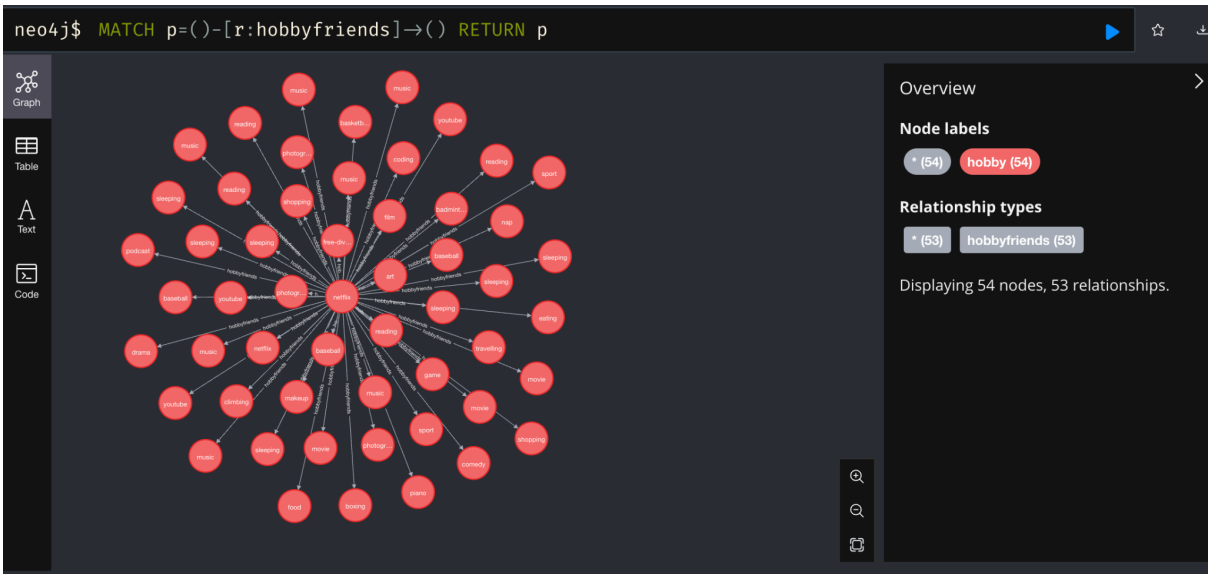


Task2

```
MATCH (me:hobby {姓名: '謝佳穎'}), (they:hobby)
WHERE they.姓名 <> '謝佳穎'
AND ANY(h IN [me.hobby1, me.hobby2, me.hobby3, me.hobby4, me.hobby5]
WHERE h IN [they.hobby1, they.hobby2, they.hobby3, they.hobby4, they.hobby5])
CREATE (me)-[r:hobbyfriends]->(they)
RETURN distinct they
```

- 選擇兩hobby node, 分別命名為 me, they

- 當me.hobby node的hobby1~hobby5與they.hobby node的hobby1~hobby5任一相同時，建立me, they兩node的關係為「hobbyfriends」。
- 共有54個與我任一hobby相同的node

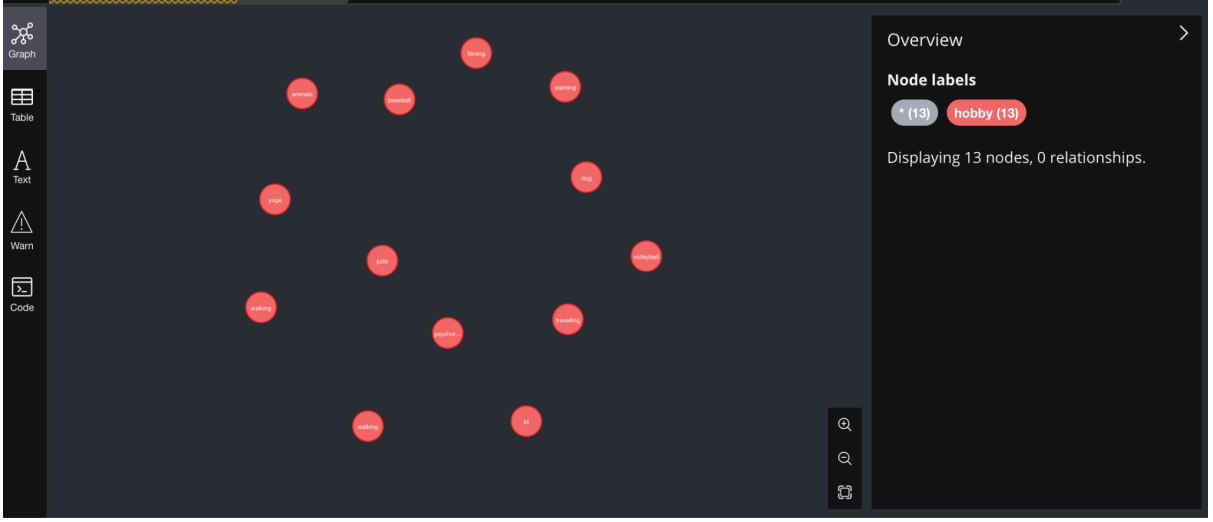


Task3

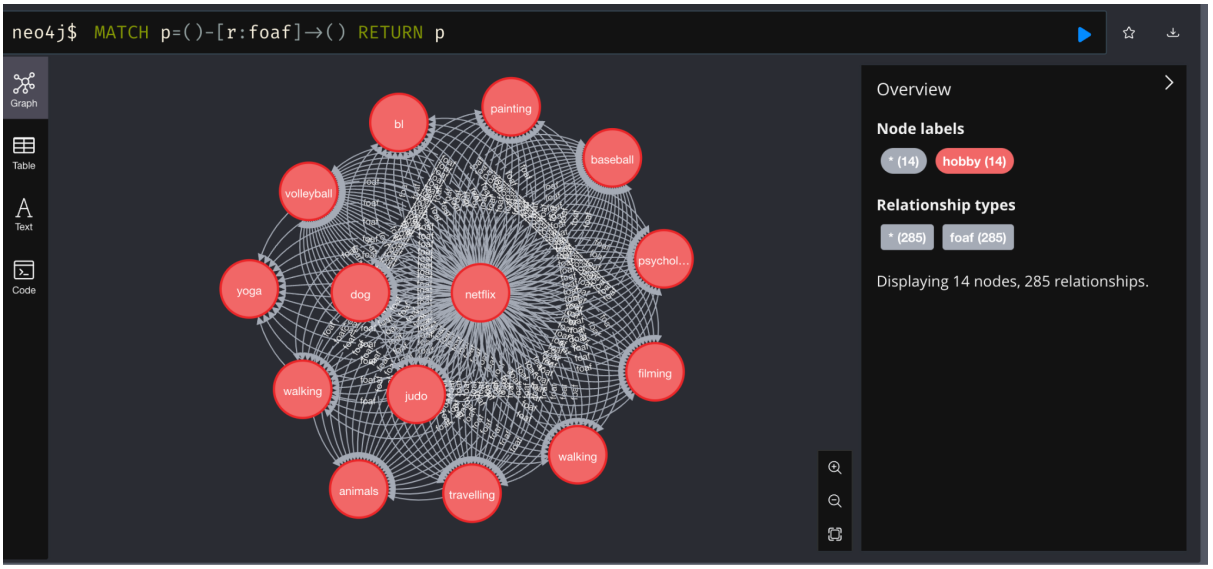
```
MATCH (me:hobby{姓名: '謝佳穎'}), (they:hobby), (theirfriend:hobby)
MATCH  (me)-[:hobbyfriends]->(they)
WHERE theirfriend.姓名 <> '謝佳穎' AND theirfriend.姓名 <> they.姓名
AND ANY(h IN [they.hobby1, they.hobby2, they.hobby3, they.hobby4, they.hobby5]
WHERE h IN [theirfriend.hobby1, theirfriend.hobby2, theirfriend.hobby3,
theirfriend.hobby4, theirfriend.hobby5])
AND NONE(h2 IN [me.hobby1, me.hobby2, me.hobby3, me.hobby4, me.hobby5]
WHERE h2 IN [theirfriend.hobby1, theirfriend.hobby2, theirfriend.hobby3,
theirfriend.hobby4, theirfriend.hobby5])
CREATE (me)-[r:foaf]->(theirfriend)
RETURN distinct theirfriend
```

- 我的hobby node命名為me
- 找到與me有hobbyfriends關係的nodes, 命名為they
- 對they與theirfriend 兩個hobby node進行篩選:
    - they的hobby1~hobby5與theirfriend的hobby1~hobby5需任一相同（theirfriend為they之hobby friend）
    - theirfriend的hobby1~hobby5不可與me的hobby1~hobby5任一相同
    - me不可是theirfriend的其中一個node
- 針對篩選出的theirfriend, 建立與me的關係，命名為foaf，挑出distinct者
- 共有13個foaf:13個人與和我任一興趣相同者任一興趣相同，且沒有任一興趣與我一樣。

```
MATCH (me:hobby{姓名: '謝佳穎'}), (they:hobby), (theirfriend:hobby)
MATCH (me)-[:hobbyfriends]→(they)
WHERE theirfriend.姓名 <> '謝佳穎' AND theirfriend.姓名 <> they.姓名
AND ANY(h IN [they.hobby1, they.hobby2, they.hobby3, they.hobby4, they.hobby5] WHERE h IN
[theirfriend.hobby1, theirfriend.hobby2, theirfriend.hobby3, theirfriend.hobby4,
theirfriend.hobby5])
AND NONE(h2 IN [me.hobby1, me.hobby2, me.hobby3, me.hobby4, me.hobby5] WHERE h2 IN
[theirfriend.hobby1, theirfriend.hobby2, theirfriend.hobby3, theirfriend.hobby4,
theirfriend.hobby5])
CREATE (me)-[r:foaf]→(theirfriend)
RETURN distinct theirfriend
```




```
neo4j$ MATCH p=()-[r:foaf]→() RETURN p
```

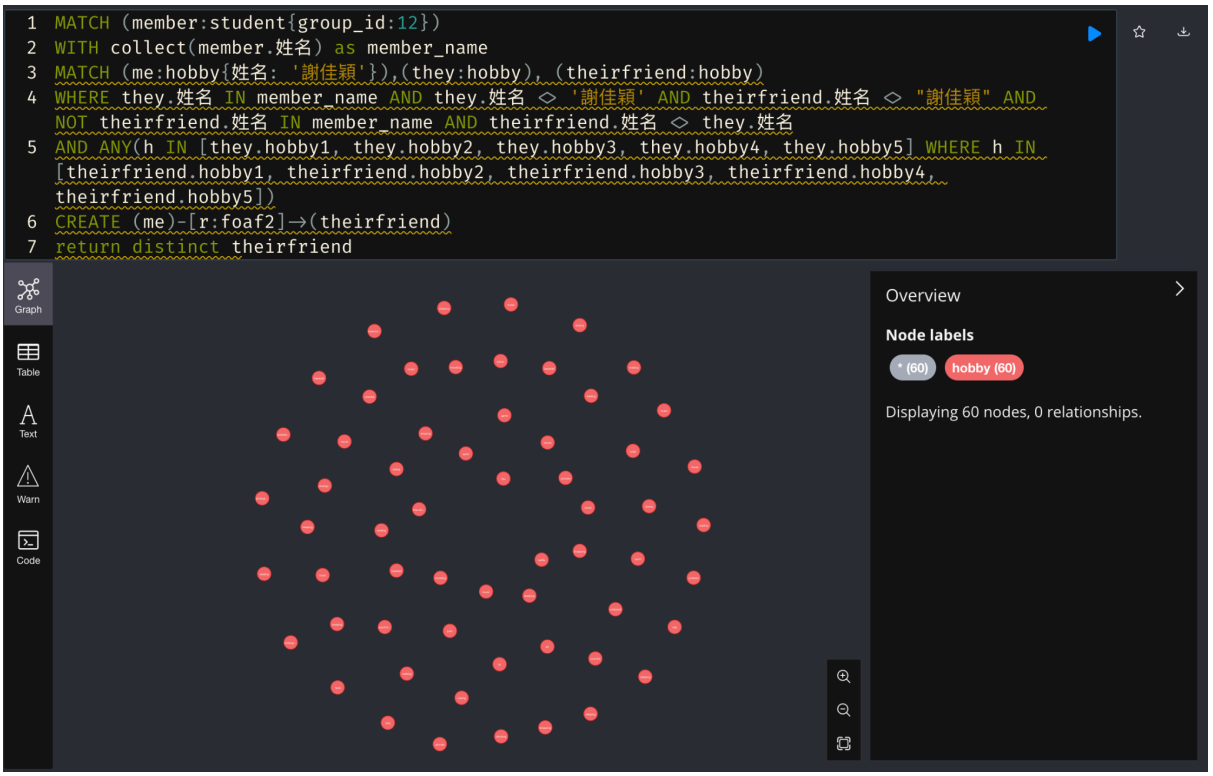


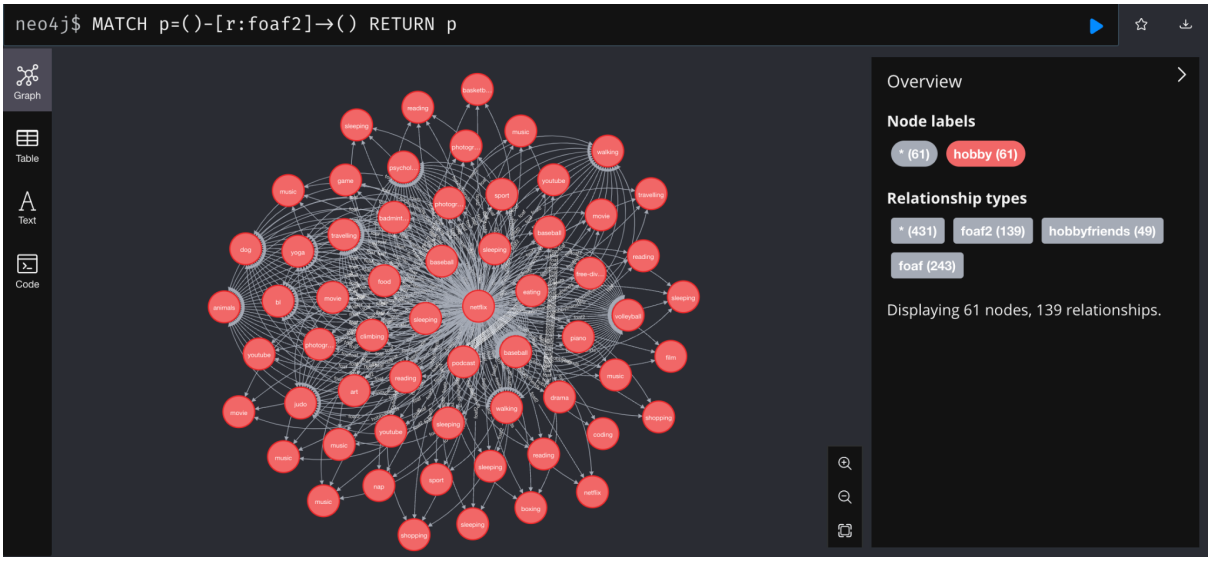
Task4

MATCH (member:student{group_id:12})
WITH collect(member.姓名) as member_name
MATCH (me:hobby{姓名: '謝佳穎'}),(they:hobby), (theirfriend:hobby)
WHERE they.姓名 IN member_name AND they.姓名 <> '謝佳穎' AND theirfriend.姓名 <> "謝佳穎" AND NOT theirfriend.姓名 IN member_name AND theirfriend.姓名 <> they.姓名
AND ANY(h IN [they.hobby1, they.hobby2, they.hobby3, they.hobby4, they.hobby5]
WHERE h IN [theirfriend.hobby1, theirfriend.hobby2, theirfriend.hobby3, theirfriend.hobby4, theirfriend.hobby5])

CREATE (me)-[r:foaf2]->(theirfriend)
return distinct theirfriend

- 我們是第12組，以group_id這個之前建立的property篩選出組員，將名字建立於member_name這個list之中
- 我的hobby node命名為me
- they node: 第12組其他組員的hobby node, theirfriend node: 第12組其他組員的hobbyfriend的hobby node
- 對they與theirfriend 兩個hobby node進行篩選：
    - they的hobby1~hobby5與theirfriend的hobby1~hobby5需任一相同（theirfriend為they之hobby friend）
- 針對篩選出的theirfriend, 建立與me的關係，命名為foaf2，挑出distinct者
- 共有60個foaf2: 除去組員，共有60個人與我的組員任一hobby相同





Task5

MATCH (h:hobby)
WHERE (h)-[:foaf2]-() AND NOT (h)-[:foaf]-() AND h.姓名 <> '謝佳穎'
RETURN distinct h