Competitions - CTFs

Marcin Kyć, Tomasz Dąbrowski

Nodes

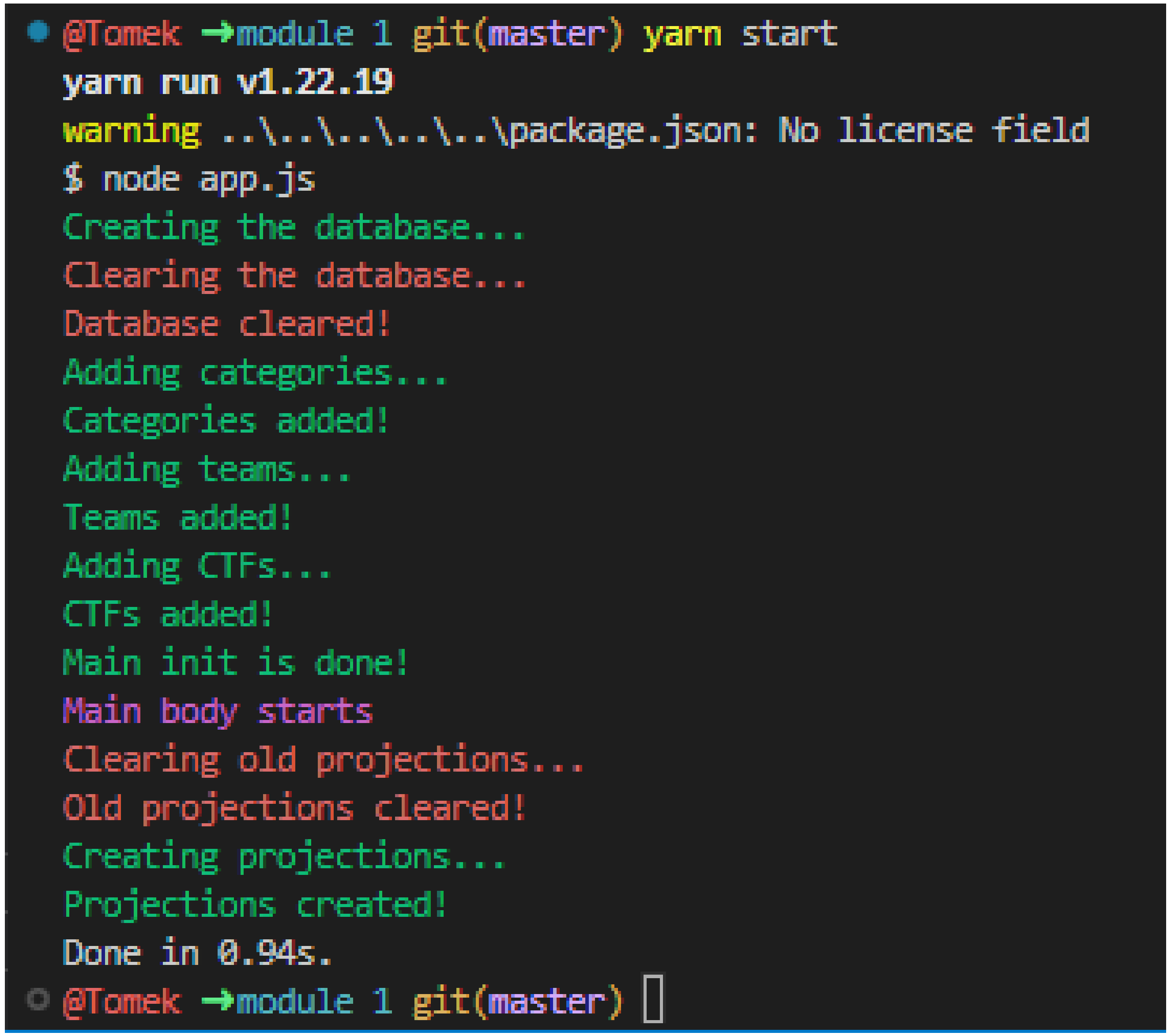
- Category – name, description,
- Challenge – name, description, points, flag, ctfName,
- Team – name,
- Member – name, email.

Relationships

- (challenge) IS_IN_CATEGORY (category),
- (member) SOLVED (challenge) – time: datetime(...),
- (member) IS_MEMBER_OF (team) – since: date(...)

Generator

Initial generation



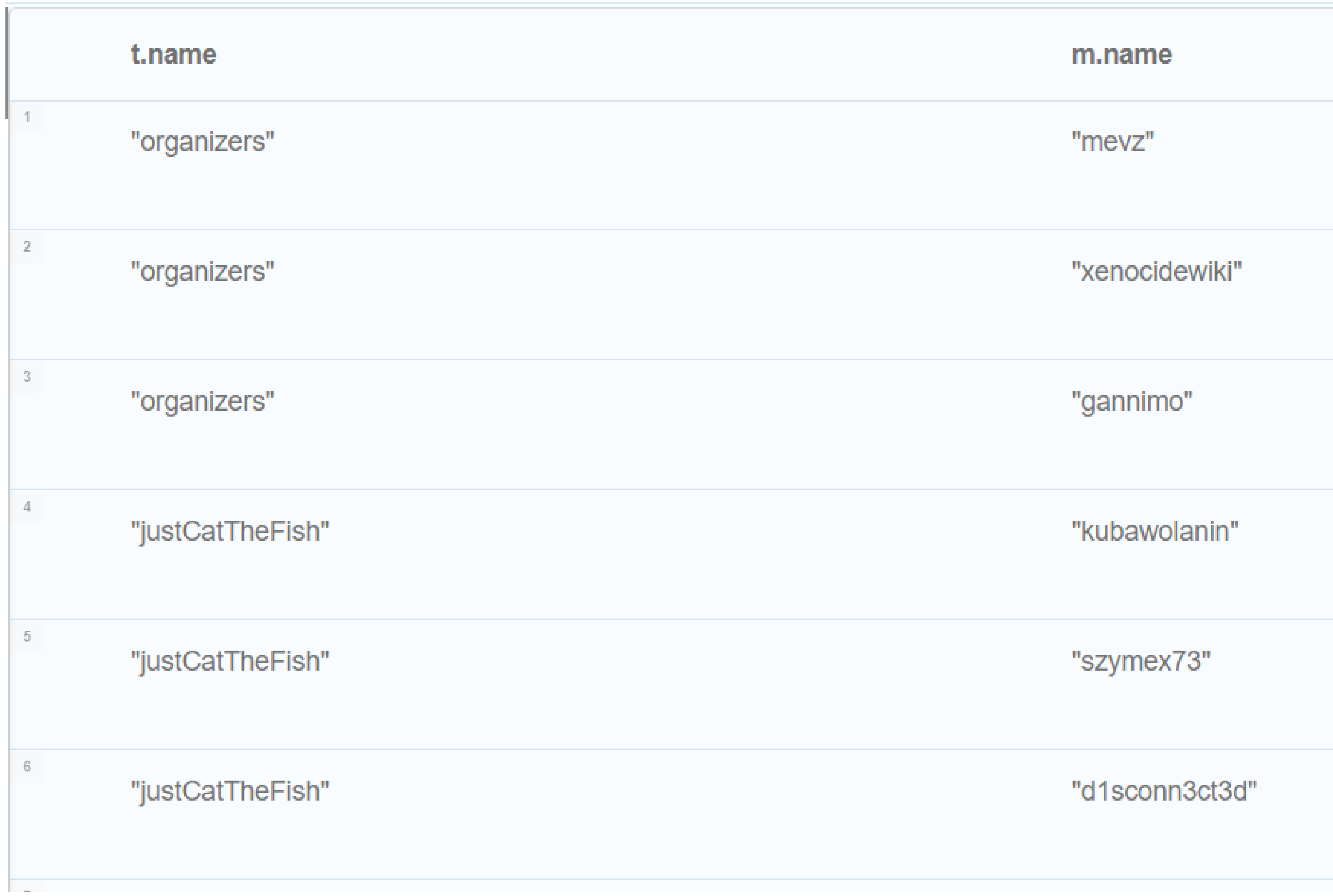
```
● @Tomek ➜module 1 git(master) yarn start
  yarn run v1.22.19
  warning ..\..\..\..\..\package.json: No license field
  $ node app.js
  Creating the database...
  Clearing the database...
  Database cleared!
  Adding categories...
  Categories added!
  Adding teams...
  Teams added!
  Adding CTFs...
  CTFs added!
  Main init is done!
  Main body starts
  Clearing old projections...
  Old projections cleared!
  Creating projections...
  Projections created!
  Done in 0.94s.
○ @Tomek ➜module 1 git(master) ▯
```

Competency questions

And their queries

1. Teams and their members

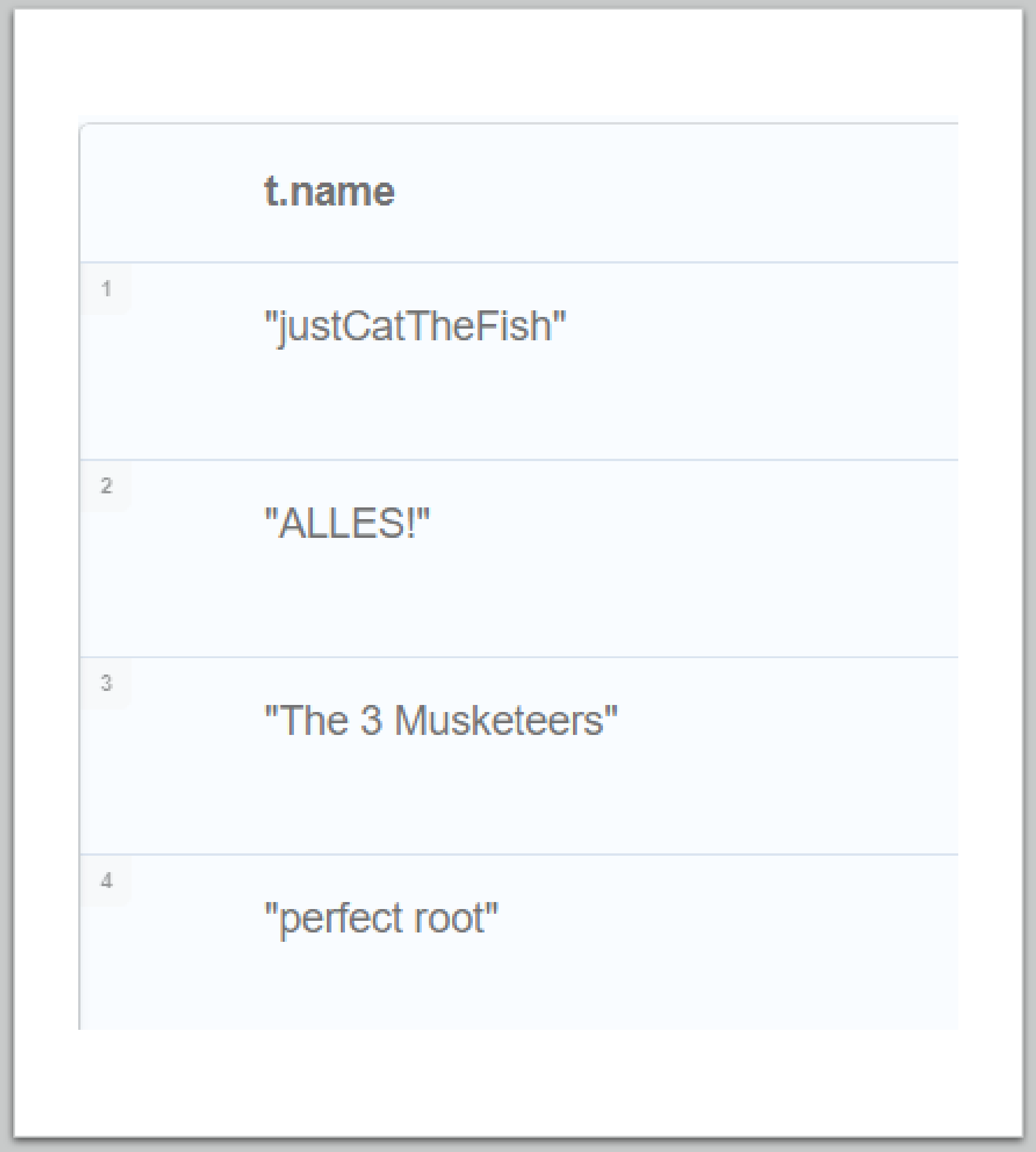
```
match (m:Member) -
[:IS_MEMBER_OF] -> (t:Team)
return t.name, m.name
```



	t.name	m.name
1	"organizers"	"mevz"
2	"organizers"	"xenocidewiki"
3	"organizers"	"gannimo"
4	"justCatTheFish"	"kubawolanin"
5	"justCatTheFish"	"szymex73"
6	"justCatTheFish"	"d1sconn3ct3d"

2. Teams that solved at least 6 tasks

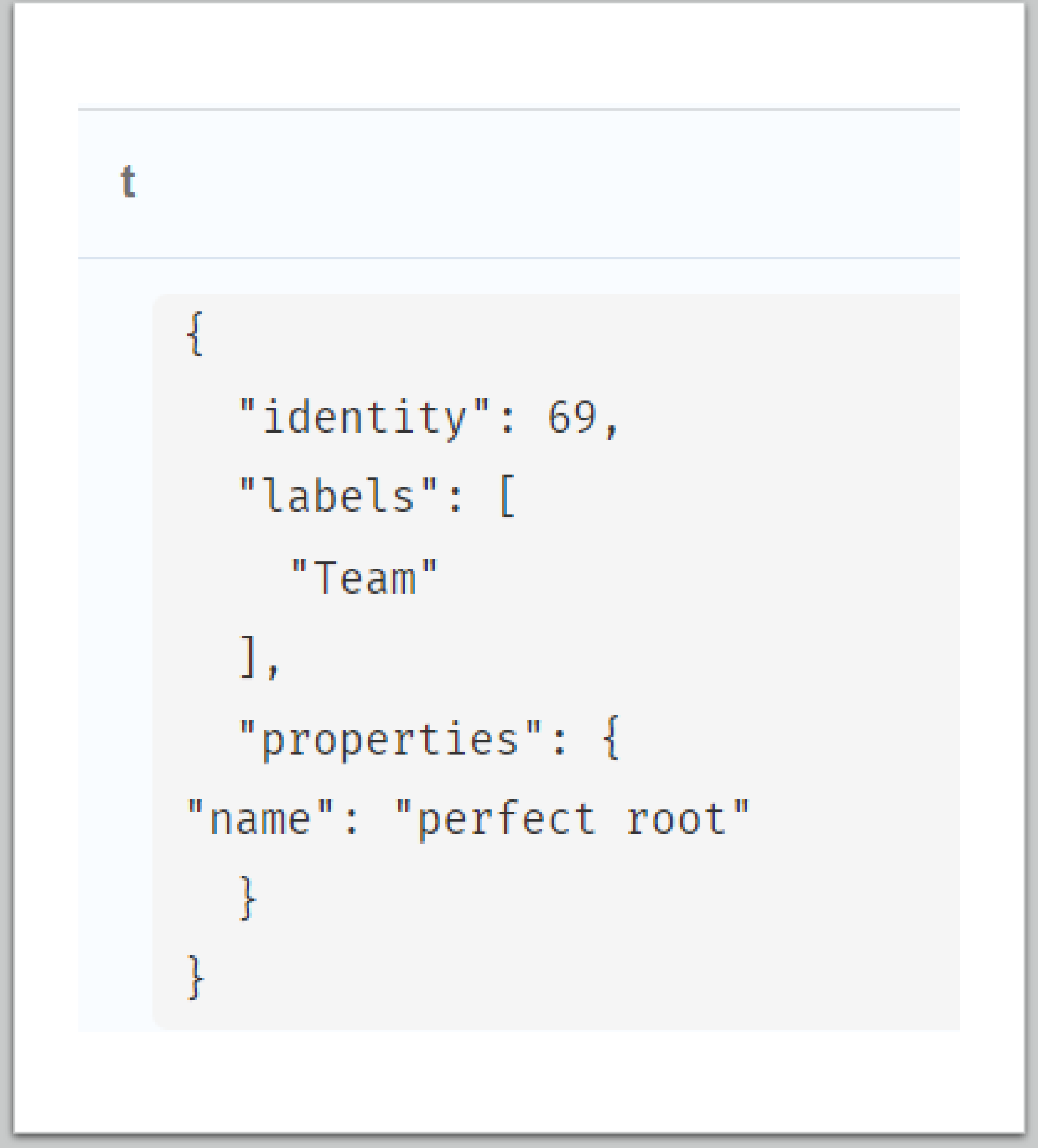
- match (:Challenge) <- [r:SOLVED] - (:Member) - [:IS_MEMBER_OF] -> (t:Team) with count(r) as cr, t where cr >= 6 return t.name



	t.name
1	"justCatTheFish"
2	"ALLES!"
3	"The 3 Musketeers"
4	"perfect root"

3. Teams that managed to solve at least one task in each category

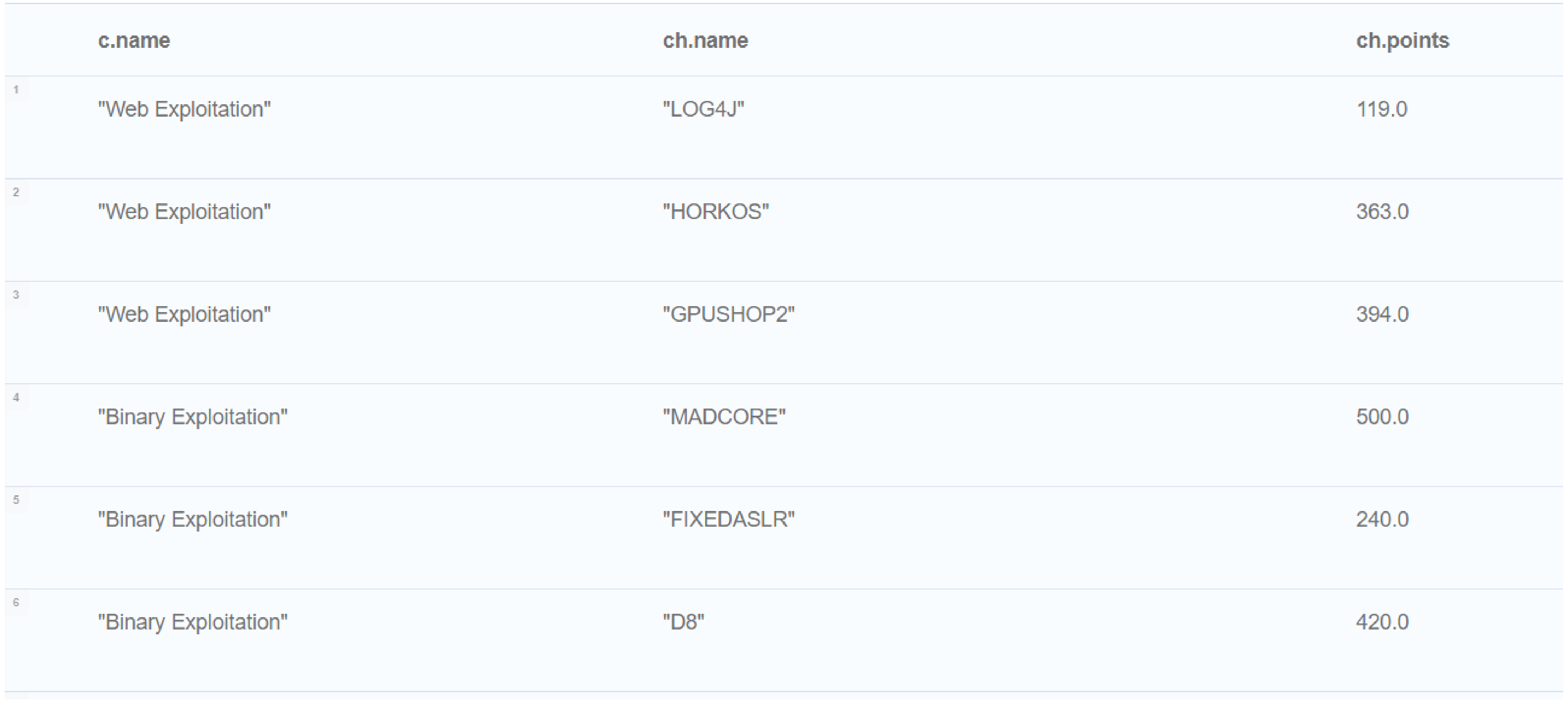
- match (t:Team) <-- (:Member) --> (:Challenge) --> (c:Category) call {match (c:Category) return count(c) as count_cat} with count(distinct c) as cc, t, count_cat where cc = count_cat return t, cc


t

```
{
  "identity": 69,
  "labels": [
    "Team"
  ],
  "properties": {
"name": "perfect root"
  }
}
```

4. Categories and their challenges

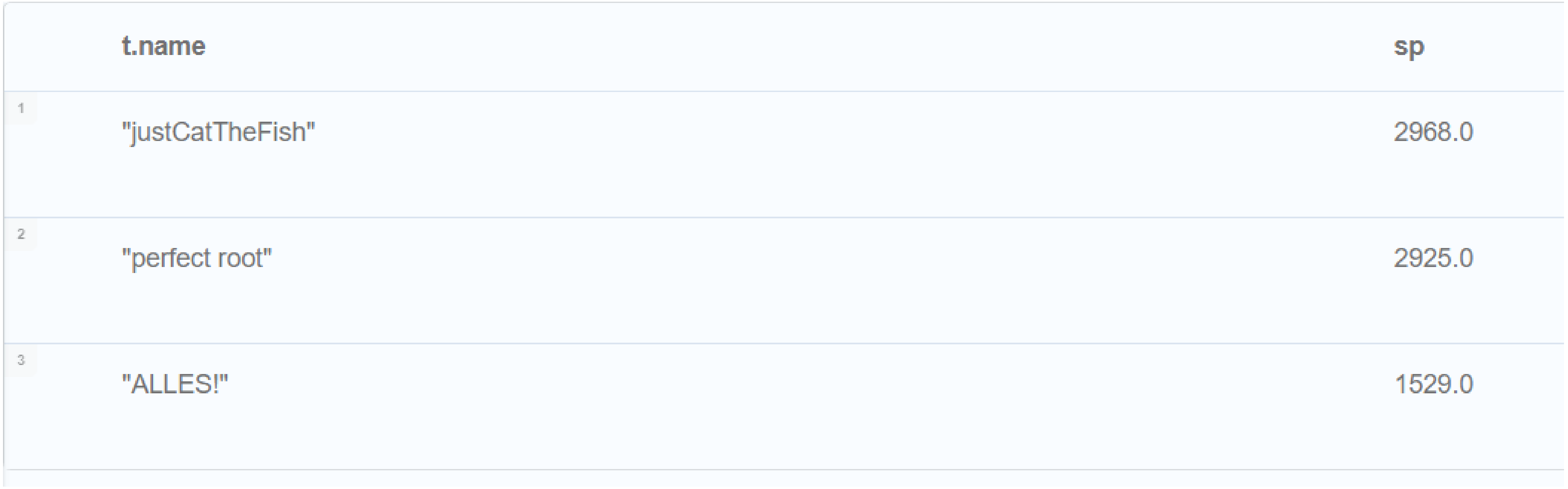
match (c:Category) <- [:IS_IN_CATEGORY] - (ch:Challenge) return c.name, ch.name, ch.points



	c.name	ch.name	ch.points
1	"Web Exploitation"	"LOG4J"	119.0
2	"Web Exploitation"	"HORKOS"	363.0
3	"Web Exploitation"	"GPUSHOP2"	394.0
4	"Binary Exploitation"	"MADCORE"	500.0
5	"Binary Exploitation"	"FIXEDASLR"	240.0
6	"Binary Exploitation"	"D8"	420.0

5. 3 teams and their members with the highest scores

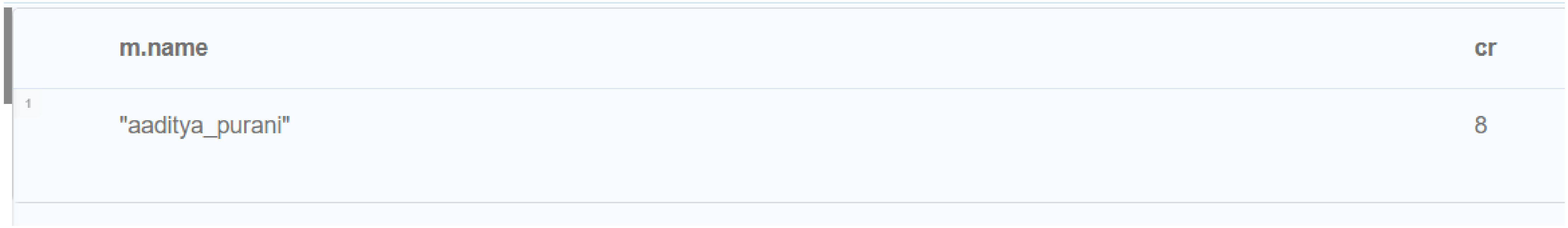
- match (t:Team) <- [:IS_MEMBER_OF] - (:Member) - [:SOLVED] -> (c:Challenge) with sum(c.points) as sp, t return t.name, sp order by sp desc limit 3



	t.name	sp
1	"justCatTheFish"	2968.0
2	"perfect root"	2925.0
3	"ALLES!"	1529.0

6. Member/s who solved the highest number of challenges

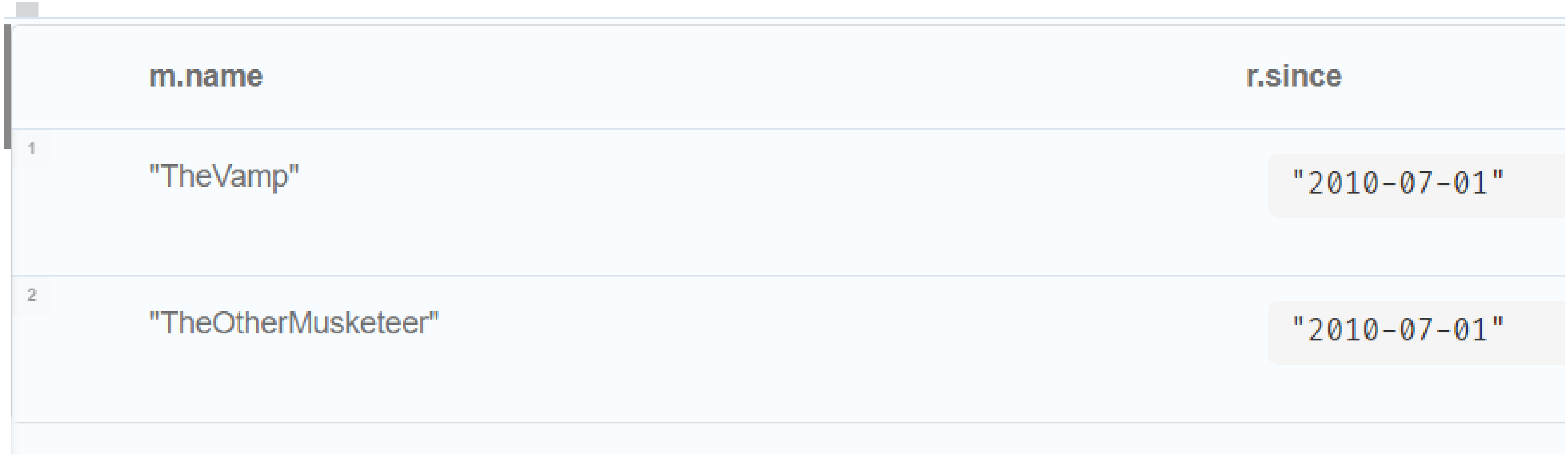
- match (m:Member) - [r:SOLVED] -> (:Challenge) call {match (m:Member) - [r:SOLVED] -> (:Challenge) with count(r) as cr, m with max(cr) as mr return mr} with count(r) as cr, m, mr where cr = mr return m.name, cr



	m.name	cr
1	"aaditya_purani"	8

7. The person/s who joined a team the longest time ago

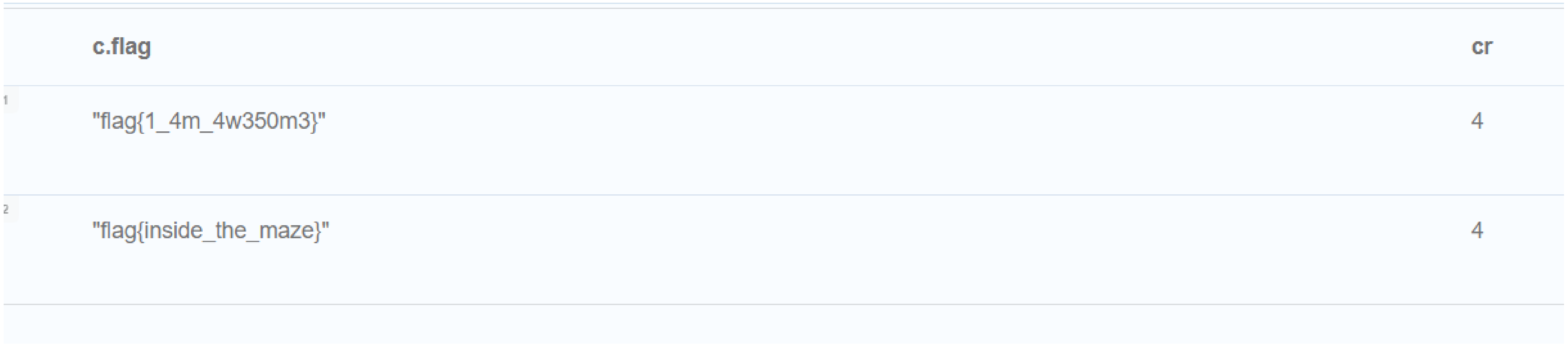
- match (m:Member) - [r:IS_MEMBER_OF] -> () call {match (m:Member) - [r:IS_MEMBER_OF] -> () with min(r.since) as ms return ms} with ms, r, m where ms = r.since return m.name, r.since



	m.name	r.since
1	"TheVamp"	"2010-07-01"
2	"TheOtherMusketeer"	"2010-07-01"

8. The flag/s of a challenge/s that was solved by the biggest number of members

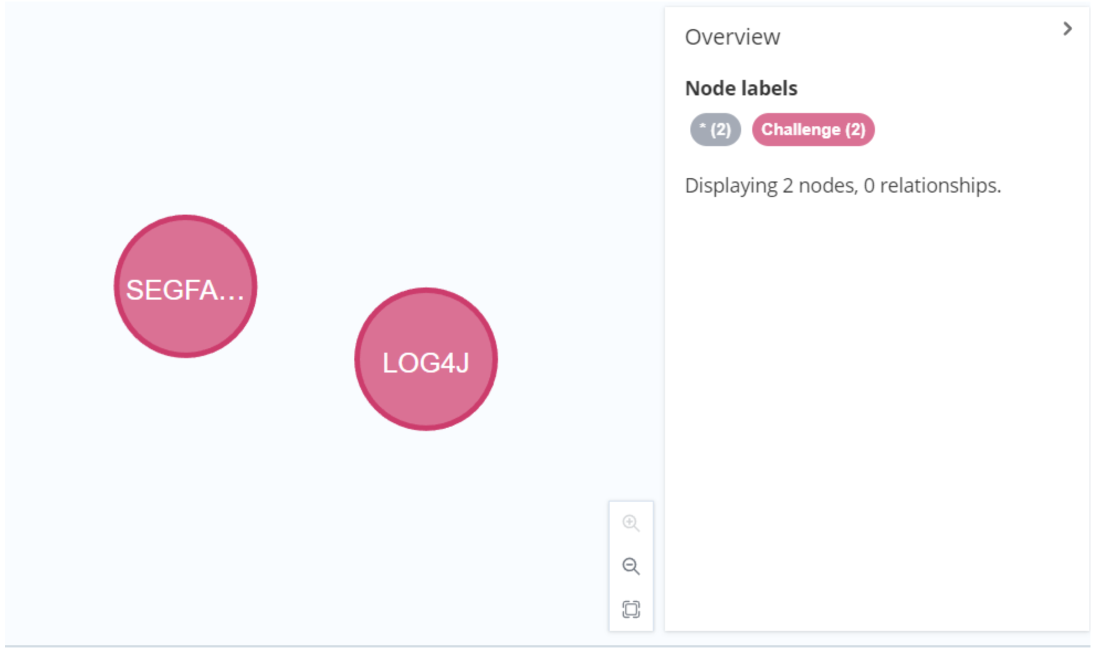
- match (:Member) - [r:SOLVED] -> (c:Challenge) call {match (:Member) - [r:SOLVED] -> (c:Challenge) with count(r) as cr, c with max(cr) as mr return mr} with count(r) as cr, c, mr where cr = mr return c.flag, cr

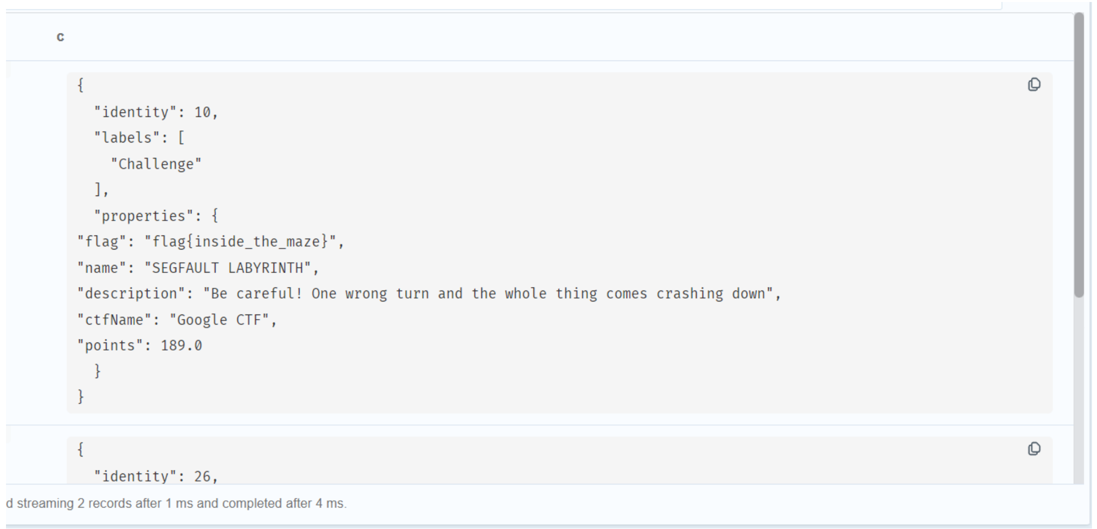


	c.flag	cr
1	"flag{1_4m_4w350m3}"	4
2	"flag{inside_the_maze}"	4

9. The challenges that were solved by the biggest number of teams and their flags

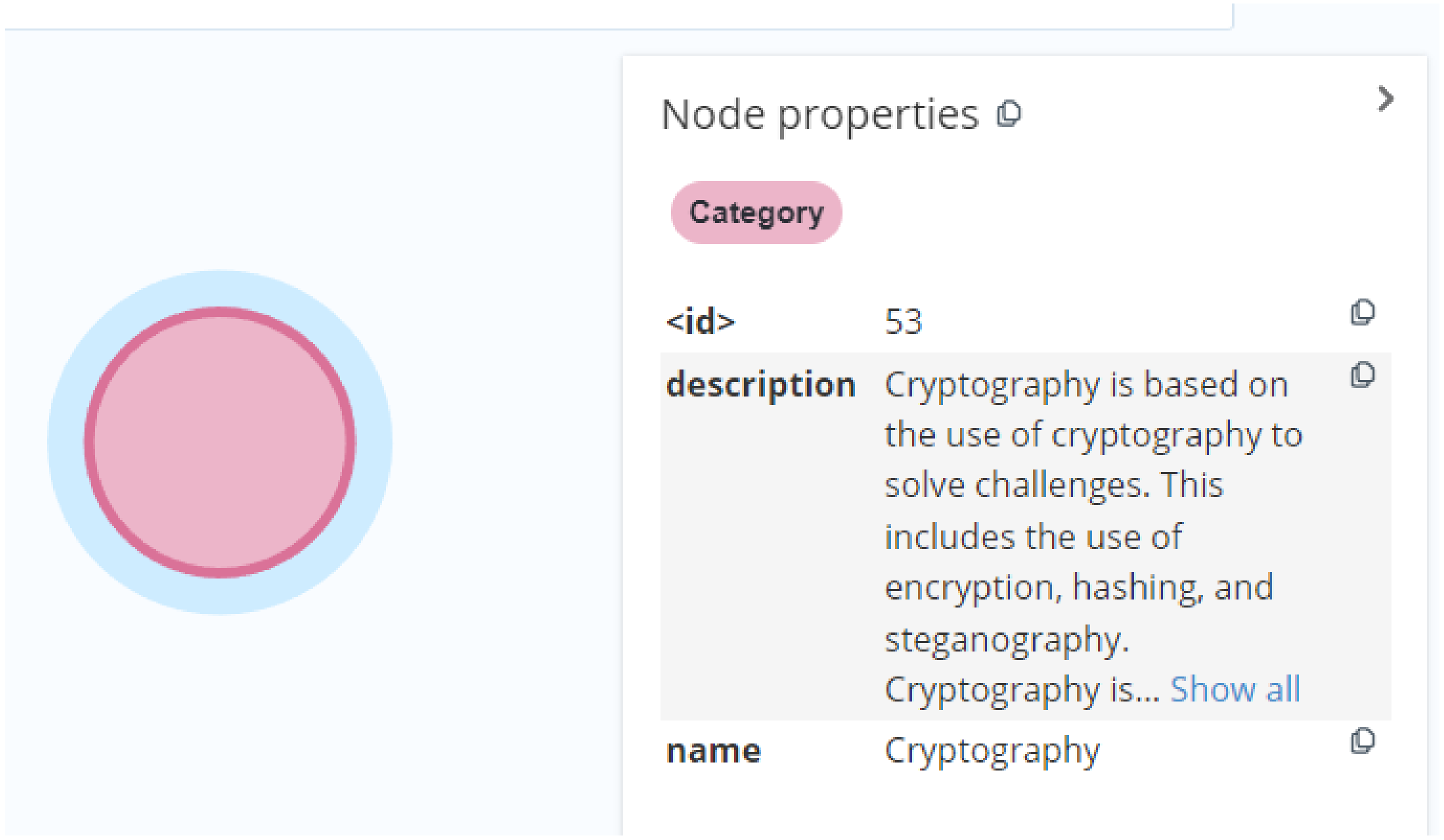
- match (t:Team) <- [:IS_MEMBER_OF] - (:Member) - [r:SOLVED] -> (c:Challenge) call {match (t:Team) <- [:IS_MEMBER_OF] - (:Member) - [r:SOLVED] -> (c:Challenge) with count(r) as cr, c with max(cr) as mc return mc} with count(r) as cr, c, mc where cr = mc return c.flag, cr




```
c
{
  "identity": 10,
  "labels": [
    "Challenge"
  ],
  "properties": {
"flag": "flag{inside_the_maze}",
"name": "SEGFAULT LABYRINTH",
"description": "Be careful! One wrong turn and the whole thing comes crashing down",
"ctfName": "Google CTF",
"points": 189.0
  }
}
{
  "identity": 26,
d streaming 2 records after 1 ms and completed after 4 ms.
```

10. Category that has the biggest sum of possible points from challenges

- match (c:Category) <- [:IS_IN_CATEGORY] - (ch:Challenge) call {match (c:Category) <- [:IS_IN_CATEGORY] - (ch:Challenge) with sum(ch.points) as sum_pts, c return max(sum_pts) as max_points} with sum(ch.points) as sum_points, c, max_points where sum_points = max_points return c, sum_points


Node properties

Category

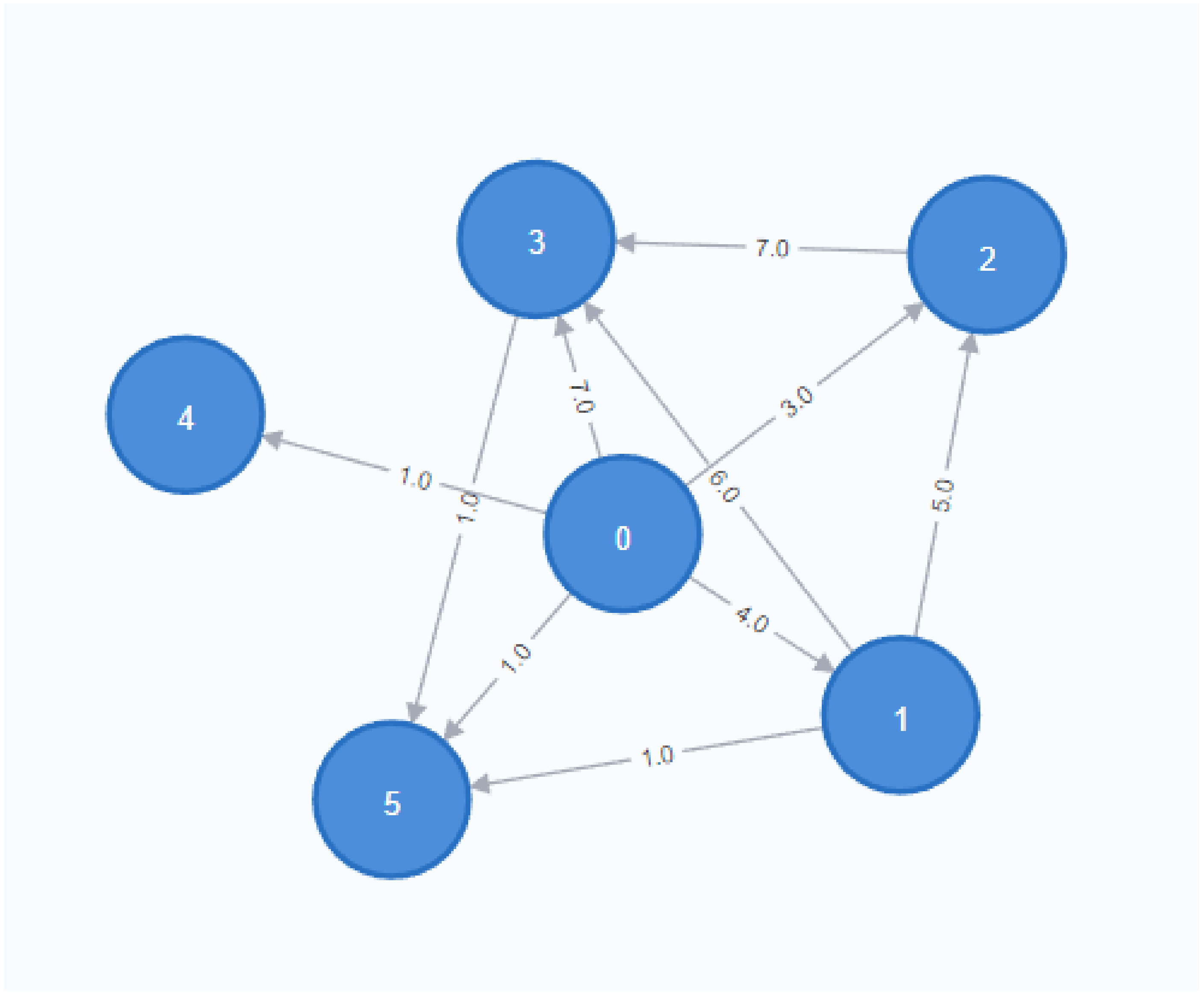
<id>	53
description	Cryptography is based on the use of cryptography to solve challenges. This includes the use of encryption, hashing, and steganography. Cryptography is... Show all
name	Cryptography

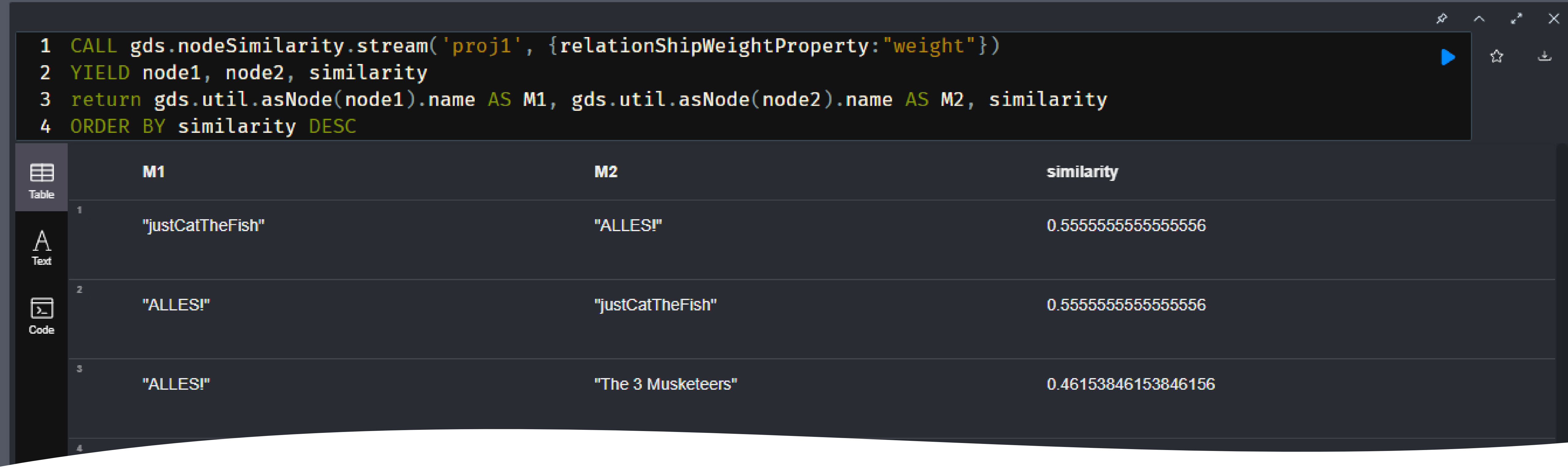
Projections

Graph analysis

1. Weights between teams that shows how many same tasks teams solved

- CALL gds.graph.project.cypher(
- 'proj1',
- 'MATCH (t:Team) RETURN id(t) AS id, labels(t) AS labels',
- 'MATCH
- (t1:Team)<--(:Member)-[:SOLVED]->(r:Challenge)<-[:SOLVED]-(:Member)-->(t2:Team)
- WHERE ID(t1) < ID(t2)
- RETURN DISTINCT id(t1) AS source, id(t2) AS target, count(r)
- AS weight'
-)





```
CALL gds.nodeSimilarity.stream('proj1', {relationShipWeightProperty:"weight"})
YIELD node1, node2, similarity
return gds.util.asNode(node1).name AS M1, gds.util.asNode(node2).name AS M2, similarity
ORDER BY similarity DESC
```

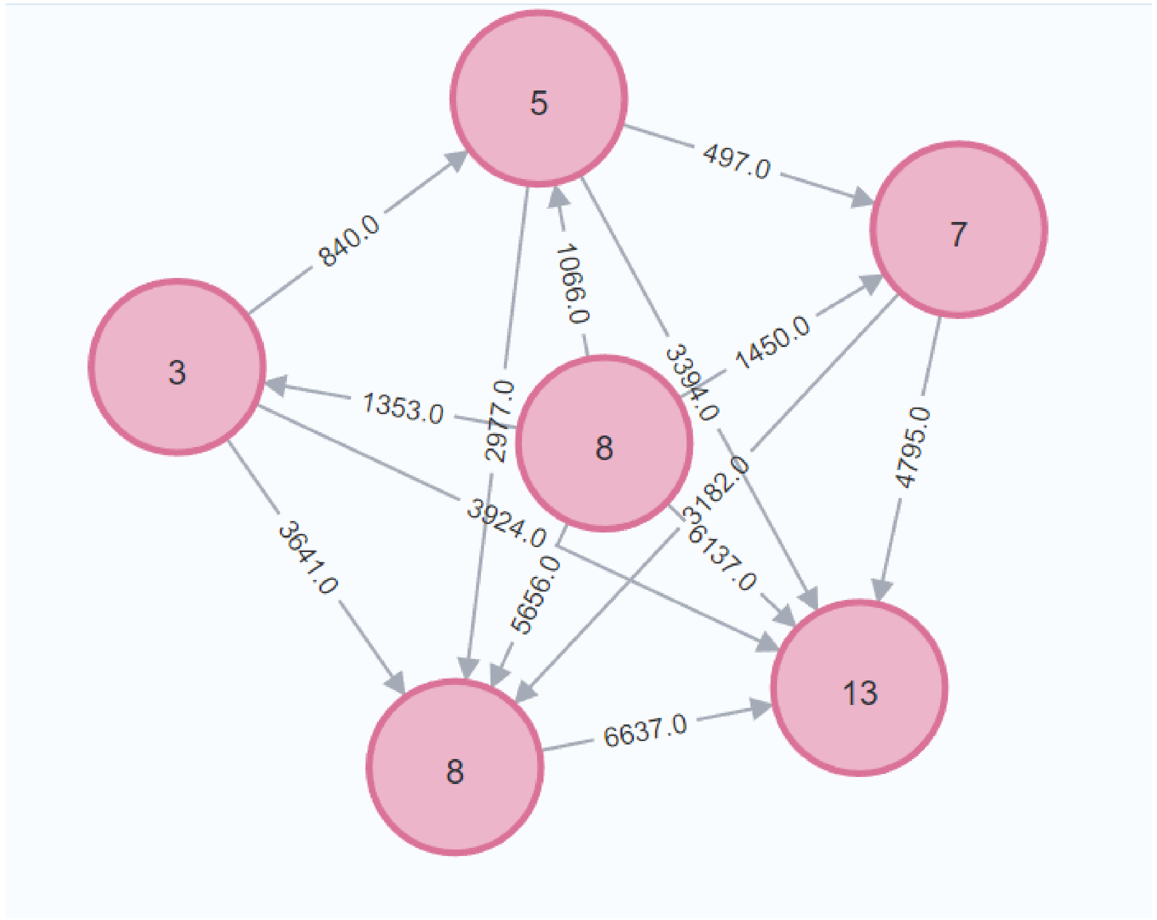
	M1	M2	similarity
1	"justCatTheFish"	"ALLES!"	0.5555555555555556
2	"ALLES!"	"justCatTheFish"	0.5555555555555556
3	"ALLES!"	"The 3 Musketeers"	0.46153846153846156
4			

1st algorithm - Node similarity - What is the similarity between each team?

- CALL gds.nodeSimilarity.stream('proj1', {relationShipWeightProperty:"weight"})
- YIELD node1, node2, similarity
- return gds.util.asNode(node1).name AS M1, gd.util.asNode(node2).name AS M2, similarity
- ORDER BY similarity DESC

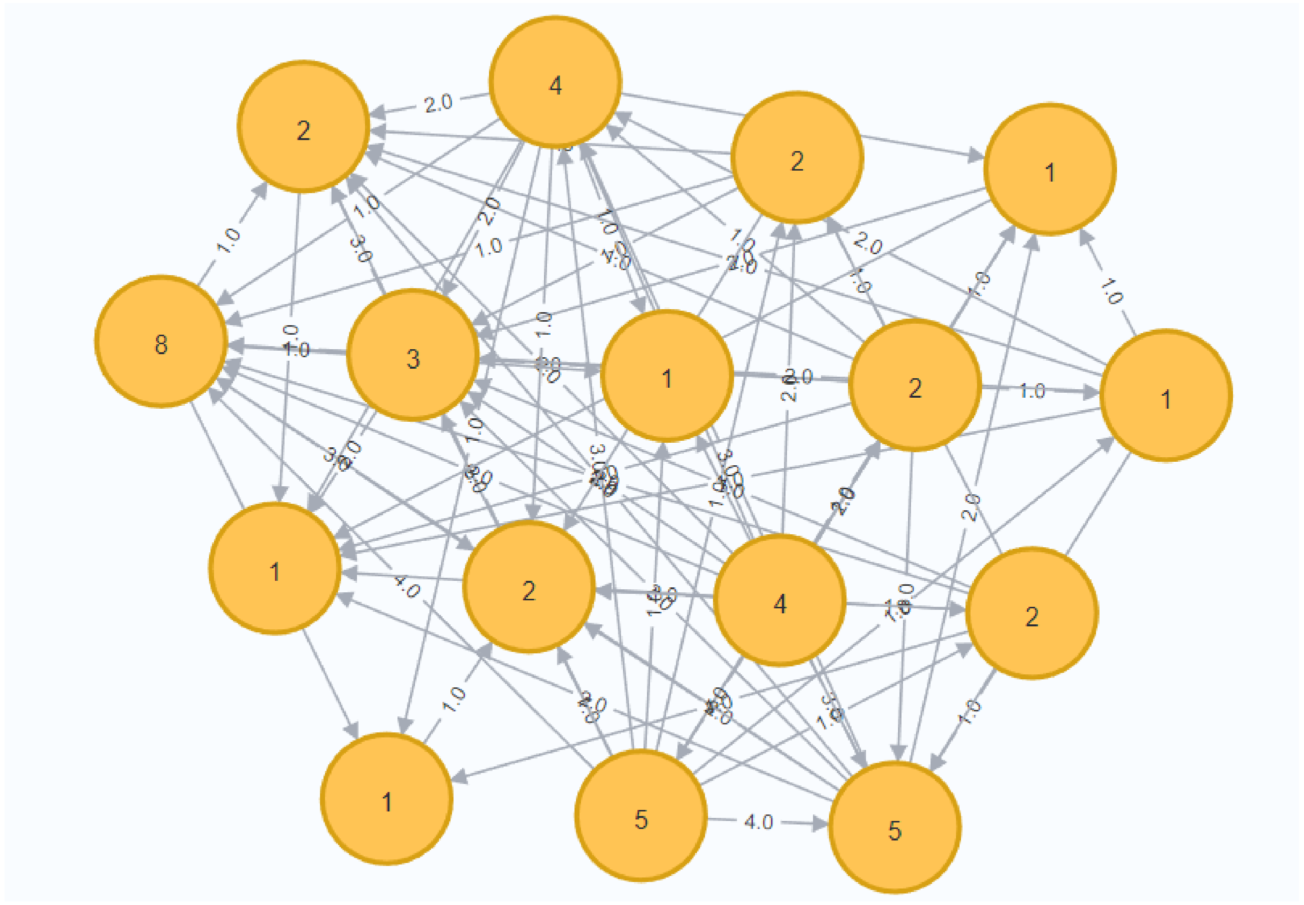
2. Categories in properties have number of solved tasks in each category by teams.
Weights represents the sum of points of challenges of two connected categories.

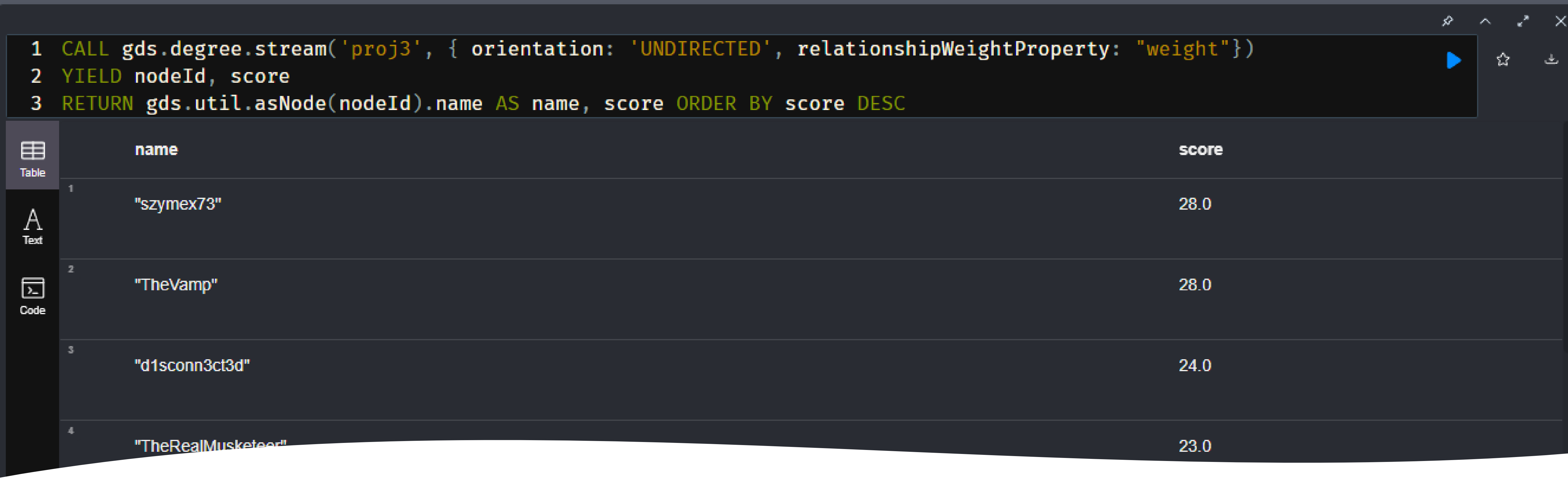
- CALL gds.graph.project.cypher(
- 'proj2',
- 'MATCH (c:Category)<--(:Challenge)<-[r:SOLVED]-(:Member)-->(t:Team) RETURN
- DISTINCT id(c) AS id, count(r) AS numberOfSolved,
- labels(c) AS labels',
- 'MATCH
- (c1:Category)<--(ch1:Challenge)<-[r1:SOLVED]-(:Member)-->(t:Team)<--(:Member)-[r2:SOLVED]->(ch2:Challenge)-->(c2:Category)
- WHERE ID(c1) < ID(c2)
- RETURN DISTINCT id(c1) AS source, id(c2) AS target,
- sum(ch1.points + ch2.points)
- AS weight')



3. Weights - in how many different categories two members have managed to solve at least one challenge. In members properties - how many challenges each managed to solve.

- CALL gds.graph.project.cypher(
- 'proj3',
- 'MATCH (:Challenge)<-[r:SOLVED]-(m:Member)-->(t:Team) RETURN
- DISTINCT id(m) AS id, count(r) AS numberOfSolved,
- labels(m) AS labels',
- 'MATCH
- (m1:Member)-[:SOLVED]->(:Challenge)-->(c:Category)<--(:Challenge)<-[:SOLVED]-(m2:Member)
- WHERE ID(m1) < ID(m2)
- RETURN DISTINCT id(m1) AS source, id(m2) AS target,
- count(c) AS weight')





```
CALL gds.degree.stream('proj3', { orientation: 'UNDIRECTED', relationshipWeightProperty: "weight"})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score ORDER BY score DESC
```

	name	score
1	"szymex73"	28.0
2	"TheVamp"	28.0
3	"d1sconn3ct3d"	24.0
4	"TheRealMusketeer"	23.0

2nd algorithm – degree centrality of players across categories and challenges

CALL gds.degree.stream('proj3', { orientation: 'UNDIRECTED', relationshipWeightProperty: "weight"})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score ORDER BY score DESC

Thank you for your attention