

All queries used in the second project:

```
Cypher(  
  ""  
  MATCH n-[r]->m  
  DELETE r, m, n  
  "").execute()
```

```
Cypher(  
  ""  
  MATCH n  
  DELETE n  
  "").execute()
```

```
Cypher(  
  ""  
  USING PERIODIC COMMIT 10000  
  LOAD CSV FROM {fileLocation} AS line  
  CREATE (uu:UserNode{UID:line[0], FName:line[1], LName:line[2]})  
  "").on("fileLocation" -> file).execute()
```

```
Cypher(  
  ""  
  MATCH (u:UserNode{UID:{header}})  
  DELETE u  
  "").on("header" -> x).execute()
```

```
Cypher(  
  ""  
  USING PERIODIC COMMIT 10000  
  LOAD CSV FROM {fileLocation} AS line  
  MATCH (u:UserNode{UID:line[0]})  
  MERGE (ss:SkillNode{Name:line[1]})  
  CREATE (u)-[r:SKILLED{Level:toFloat(line[2])}]->(ss)  
  "").on("fileLocation" -> file).execute()
```

```
Cypher(  
  ""  
  USING PERIODIC COMMIT 10000  
  LOAD CSV FROM {fileLocation} AS line  
  MATCH (u:UserNode{UID:line[0]})  
  MERGE (ee:InterestNode {Name:line[1]})  
  CREATE (u)-[r:INTERESTED{Level:toFloat(line[2])}]->(ee)  
  "").on("fileLocation" -> file).execute()
```

```
Cypher(  
  ""
```

```

USING PERIODIC COMMIT 10000
LOAD CSV FROM {fileLocation} AS line
MATCH (u:UserNode{UID:line[0]})
MERGE (pp:ProjectNode {PName:line[1]})
CREATE (u)-[r:WORKS_ON]->(pp)
""").on("fileLocation" -> file).execute()

```

*Cypher*(  
 """)

```

USING PERIODIC COMMIT 10000
LOAD CSV FROM {fileLocation} AS line
MATCH (u:UserNode{UID:line[0]})
MERGE (oo:OrganizationNode {OName:line[1], OType:line[2]})
CREATE (u)-[r:BELONGS_TO]->(oo)
""").on("fileLocation" -> file).execute()

```

*Cypher*(  
 """)

```

USING PERIODIC COMMIT 10000
LOAD CSV FROM {fileLocation} AS line
MATCH
(o1:OrganizationNode{OName:line[0]}),(o2:OrganizationNode{OName:line[1]})
CREATE (o1)-[r:DISTANCE_TO]{Distance:toFloat(line[2])}->(o2)
""").on("fileLocation" -> file).execute()

```

**val** comm = *Cypher*(  
 """)

```

MATCH (oo:OrganizationNode), ((o:OrganizationNode)-[d:DISTANCE_TO]-
(userOrg:OrganizationNode{OType:UPPER({type}})), ((u:UserNode)-
[r:INTERESTED|SKILLED]-(is)-[ur]-(user:UserNode{UID:{x}}))
WHERE (user <> u) AND (user-->userOrg) AND (d.Distance <= {y}) AND ((u--
>o) OR (u-->userOrg)) AND (u-->is<--user) AND (u-->oo)
RETURN "User: " + u.UID + "; Organization: " + oo.OName + "; Weight: " as
ido, is.Name as isName, r.Level as level, ur.Level as ulevel
""").on("x" -> user, "y" -> distance, "type" -> organizationType)

```

**val** comm = *Cypher*(  
 """)

```

UNWIND {myList} as partInt
MATCH (user:UserNode{UID:{x}}), (col:UserNode), (colOfCol:UserNode),
(p1:ProjectNode), (p2:ProjectNode), (i:InterestNode{Name:UPPER(partInt)})
WHERE (user<>col) AND ((user)-->(p1)<--(col)-->(p2)<--(colOfCol)) AND
(colOfCol-->i)
RETURN colOfCol.FName as firstName, colOfCol.LName as lastName,
count(colOfCol.UID) as counter
""").on("x" -> user, "myList" -> particularInterests)

```