Part 1

Dump file link
https://cloud.technikum-wien.at/s/cr2e8mKFQxHEyrr

Preparation

Execute import command for newfile_01.csv (x10, once for each file)

```
LOAD CSV WITH HEADERS FROM "file:///newfile_01.csv" AS row
MERGE (u:User {userId: substring(row.userID, 2)}) WITH *
MATCH (m:Movie {imdbId: row.imdbID})
MERGE (u)-[:RATED {rating: toFloat(row.rating), timestamp:
apoc.date.parse(row.review_date, "s", "dd MMMMM yyyy")}]->(m)
```

Tasks

We need a "User" for ratings

```
CREATE (n:User {userId:'99999999999', name: 'Dominik & Ruben'})
```

Select the new user

```
MATCH (n:User)
WHERE n.name = 'Dominik & Ruben'
RETURN n
```

or

```
MATCH (u:User)
WHERE u.userId = '99999999999'
RETURN u
```

Create ratings

Titanic(1721) 5*

```
MATCH (u:User {userId: '99999999999'})
MATCH (m:Movie {movieId: '1721'})
CREATE (u)-[:RATED {rating: 5.0, timestamp: timestamp()}]->(m)
```

Home Alone (586) 5*

```
MATCH (u:User {userId: '99999999999'})
MATCH (m:Movie {movieId: '586'})
CREATE (u)-[:RATED {rating: 5.0, timestamp: timestamp()}]->(m)
```

Deadpool (122904) 4*

```
MATCH (u:User {userId: '99999999999'})
MATCH (m:Movie {movieId: '122904'})
CREATE (u)-[:RATED {rating: 4.0, timestamp: timestamp()}]->(m)
```

Beerfest (47640) 4*

```
MATCH (u:User {userId: '99999999999'})
MATCH (m:Movie {movieId: '47640'})
CREATE (u)-[:RATED {rating: 4.0, timestamp: timestamp()}]->(m)
```

Scary Movie (3785) 4*

```
MATCH (u:User {userId: '99999999999'})
MATCH (m:Movie {movieId: '3785'})
CREATE (u)-[:RATED {rating: 4.0, timestamp: timestamp()}]->(m)
```

Fast and the Furious (46335) 6.8*

```
MATCH (u:User {userId: '99999999999'})
MATCH (m:Movie {movieId: '46335'})
CREATE (u)-[:RATED {rating: 6.8, timestamp: timestamp()}]->(m)
```

Get rated movies of user

```
MATCH (u:User {userId: '99999999999'})-[:RATED]->(m:Movie)
RETURN u, m
```

Get users which also rated a movie similar to me using EUCLIDEAN DISTANCE

```
MATCH (me:User {userId: '99999999999'})-[myRating:RATED]->(m:Movie)<-
[otherRating:RATED]-(other:User)
WITH me, other, COLLECT(myRating.rating) AS
myRatings,COLLECT(otherRating.rating) AS othersRating, COLLECT(m) AS
movies
WHERE other <> me
WITH me, other, gds.similarity.euclideanDistance(myRatings,
othersRating) AS similarity,myRatings, othersRating, movies
ORDER BY similarity ASC
LIMIT 5
RETURN other, movies, similarity, myRatings, othersRating
```

Get users which also rated a movie similar to me using EUCLIDEAN
Desc in that case

```
MATCH (me:User {userId: '99999999999'})-[myRating:RATED]->(m:Movie)<-
[otherRating:RATED]-(other:User)
WITH me, other, COLLECT(myRating.rating) AS
myRatings,COLLECT(otherRating.rating) AS othersRating, COLLECT(m) AS
movies
WHERE other <> me
WITH me, other, gds.similarity.euclidean(myRatings, othersRating) AS
similarity,myRatings, othersRating, movies
ORDER BY similarity DESC
LIMIT 5
RETURN other, movies, similarity, myRatings, othersRating
```

Get users which also rated a movie similar to me using PEARSON
Desc in that case

```
MATCH (me:User {userId: '99999999999'})-[myRating:RATED]->(m:Movie)<-
[otherRating:RATED]-(other:User)
WITH me, other, COLLECT(myRating.rating) AS
myRatings,COLLECT(otherRating.rating) AS othersRating, COLLECT(m) AS
movies
WHERE other <> me
WITH me, other, gds.similarity.pearson(myRatings, othersRating) AS
similarity,myRatings, othersRating, movies
ORDER BY similarity DESC
LIMIT 5
RETURN other, movies, similarity, myRatings, othersRating
```

Get users which also rated a movie similar to me using OVERLAP
Desc in that case

```
MATCH (me:User {userId: '99999999999'})-[myRating:RATED]->(m:Movie)<-
[otherRating:RATED]-(other:User)
WITH me, other, COLLECT(myRating.rating) AS
myRatings,COLLECT(otherRating.rating) AS othersRating, COLLECT(m) AS
movies
WHERE other <> me
WITH me, other, gds.similarity.overlap(myRatings, othersRating) AS
similarity,myRatings, othersRating, movies
ORDER BY similarity DESC
LIMIT 5
RETURN other, movies, similarity, myRatings, othersRating
```

Get users which also rated a movie similar to me using JACCARD
Desc in that case

```
MATCH (me:User {userId: '99999999999'})-[myRating:RATED]->(m:Movie)<-
[otherRating:RATED]-(other:User)
WITH me, other, COLLECT(myRating.rating) AS
myRatings,COLLECT(otherRating.rating) AS othersRating, COLLECT(m) AS
movies
WHERE other <> me
WITH me, other, gds.similarity.jaccard(myRatings, othersRating) AS
similarity,myRatings, othersRating, movies
ORDER BY similarity DESC
LIMIT 5
RETURN other, movies, similarity, myRatings, othersRating
```

Get users which also rated a movie similar to me using COSINE
Desc in that case

```
MATCH (me:User {userId: '99999999999'})-[myRating:RATED]->(m:Movie)<-
[otherRating:RATED]-(other:User)
WITH me, other, COLLECT(myRating.rating) AS
myRatings,COLLECT(otherRating.rating) AS othersRating, COLLECT(m) AS
movies
WHERE other <> me
WITH me, other, gds.similarity.cosine(myRatings, othersRating) AS
similarity,myRatings, othersRating, movies
ORDER BY similarity DESC
LIMIT 5
RETURN other, movies, similarity, myRatings, othersRating
```

With those user ids we can see which movies the other user also rated highly.
Get all users the user rated with > 8.0

```
MATCH (u:User {userId: '<similar user id>'})-[r:RATED]->(m:Movie)
where r.rating > 8.0
RETURN u, m
```

Part 2

Preparation

Download VirtualBox and Sparql.ova

Follow the instructions from https://moodle.technikum-wien.at/pluginfile.php/2031623/mod_folder/content/0/Lesson-5-RDF-and-SPARQL.pdf?forcedownload=1

Open the interface

navigate to http://localhost:9090/sparql

Tasks

Prerequisites

Create <SocialNetwork> graph

```
CREATE GRAPH <SocialNetwork>
```

Insert the provided data

```
INSERT DATA {GRAPH <SocialNetwork> {
  <http://example.neo4j/john> <http://www.lib.org/schema#firstName>
"John" .
  <http://example.neo4j/john> <http://www.lib.org/schema#lastName>
"Cook" .
  <http://example.neo4j/john> <http://www.lib.org/schema#gender> "male"
.
  <http://example.neo4j/john> <http://www.lib.org/schema#country>
"Chile" .
  <http://example.neo4j/julie> <http://www.lib.org/schema#firstName>
"Julie" .
  <http://example.neo4j/julie> <http://www.lib.org/schema#lastName>
"Freud" .
  <http://example.neo4j/julie> <http://www.lib.org/schema#country>
"Chile" .
  <http://example.neo4j/peter> <http://www.lib.org/schema#firstName>
"Peter" .
  <http://example.neo4j/peter> <http://www.lib.org/schema#lastName>
"Norvig" .
  <http://example.neo4j/peter> <http://www.lib.org/schema#country> "US"
.
  <http://example.neo4j/peter> <http://www.lib.org/schema#gender> "male"
.
  <http://example.neo4j/mary> <http://www.lib.org/schema#firstName>
"Mary" .
```

```
  <http://example.neo4j/mary> <http://www.lib.org/schema#lastName>
"Lindt" .
  <http://example.neo4j/mary> <http://www.lib.org/schema#country>
"Sweden" .
  <http://example.neo4j/mary> <http://www.lib.org/schema#gender>
"female" .
  <http://example.neo4j/mary> <http://www.lib.org/schema#knows>
<http://example.neo4j/peter> .
  <http://example.neo4j/john> <http://www.lib.org/schema#knows>
<http://example.neo4j/julie> .
  <http://example.neo4j/julie> <http://www.lib.org/schema#knows>
<http://example.neo4j/john> .
  <http://example.neo4j/julie> <http://www.lib.org/schema#follows>
<http://example.neo4j/john> .
  <http://example.neo4j/john> <http://www.lib.org/schema#follows>
<http://example.neo4j/peter> .
  <http://example.neo4j/peter> <http://www.lib.org/schema#follows>
<http://example.neo4j/mary> .
  <http://example.neo4j/mary> <http://www.lib.org/schema#follows>
<http://example.neo4j/julie> .
  <http://example.neo4j/peter> <http://www.lib.org/schema#dislikes>
<http://example.neo4j/john> .
  <http://example.neo4j/john> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Person> .
  <http://example.neo4j/julie> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Person> .
  <http://example.neo4j/peter> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Person> .
  <http://example.neo4j/mary> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Person> .
  <http://example.neo4j/p1> <http://www.lib.org/schema#content> "I love
U2" .
  <http://example.neo4j/p1> <http://www.lib.org/schema#language> "Chile"
.
  <http://example.neo4j/p2> <http://www.lib.org/schema#content> "Queen
is awesome" .
  <http://example.neo4j/p3> <http://www.lib.org/schema#content> "I hate
U2" .
  <http://example.neo4j/p1> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Post> .
  <http://example.neo4j/p2> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Post> .
  <http://example.neo4j/p3> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Post> .
  <http://example.neo4j/t1> <http://www.lib.org/schema#content> "U2" .
  <http://example.neo4j/t1> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Tag> .
  <http://example.neo4j/t2> <http://www.lib.org/schema#content> "Queen"
.
  <http://example.neo4j/t2> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Tag> .
  <http://example.neo4j/t3> <http://www.lib.org/schema#content> "U2" .
  <http://example.neo4j/t3> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Tag> .
```

```
  <http://example.neo4j/p1> <http://www.lib.org/schema#hasTag>
<http://example.neo4j/t1> .
  <http://example.neo4j/p2> <http://www.lib.org/schema#hasTag>
<http://example.neo4j/t2> .
  <http://example.neo4j/p3> <http://www.lib.org/schema#hasTag>
<http://example.neo4j/t1> .
  <http://example.neo4j/julie> <http://www.lib.org/schema#likes>
<http://example.neo4j/p1> .
  <http://example.neo4j/john> <http://www.lib.org/schema#likes>
<http://example.neo4j/p1> .
  <http://example.neo4j/john> <http://www.lib.org/schema#dislikes>
<http://example.neo4j/p2> .
  <http://example.neo4j/peter> <http://www.lib.org/schema#dislikes>
<http://example.neo4j/p3> .
  <http://example.neo4j/john> <http://www.lib.org/schema#dislikes>
<http://example.neo4j/p3> .
  <http://example.neo4j/julie> <http://www.lib.org/schema#dislikes>
<http://example.neo4j/p3> .
  <http://example.neo4j/peter> <http://www.lib.org/schema#likes>
<http://example.neo4j/p2> .
}}
```

Check the inserted triplets

```
SELECT *
FROM <SocialNetwork>
WHERE { ?s ?p ?o }
```

Queries

Query: "Total number of likes given to posts mentioning U2, or dislikes to posts mentioning Queen"

```
PREFIX lib: <http://www.lib.org/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?post (COUNT(?action) AS ?qty)
WHERE {
  {
    ?post rdf:type foaf:Post .
    ?post lib:hasTag ?tag1 .
    ?tag1 rdf:type foaf:Tag .
    ?tag1 lib:content "U2" .
    ?action lib:likes ?post .
  }
  UNION
  {
    ?post rdf:type foaf:Post .
    ?post lib:hasTag ?tag2 .
    ?tag2 rdf:type foaf:Tag .
    ?tag2 lib:content "Queen" .
    ?action lib:dislikes ?post .
  }
}
GROUP BY ?post
```

Query: “Name, last name and gender of people who follow people who dislike at least 2 different posts, or who liked at least one post that mentions U2.”

```
PREFIX lib: <http://www.lib.org/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?fn ?ln ?g
WHERE {
  ?follower lib:follows ?target .
  ?target lib:dislikes ?post1 .
  ?post1 rdf:type foaf:Post .
  ?post1 lib:hasTag ?tag1 .
  ?tag1 rdf:type foaf:Tag .
  ?tag1 lib:content "U2" .
  {
    SELECT ?follower (COUNT(DISTINCT ?post) AS ?dislikedPostsCount)
    WHERE {
      ?follower lib:follows ?target2 .
      ?target2 lib:dislikes ?post .
      ?post rdf:type foaf:Post .
    }
    GROUP BY ?follower
    HAVING (?dislikedPostsCount >= 2)
  }
  UNION
  {
    ?follower lib:likes ?post2 .
    ?post2 rdf:type foaf:Post .
    ?post2 lib:hasTag ?tag2 .
    ?tag2 rdf:type foaf:Tag .
    ?tag2 lib:content "U2" .
  }
  ?follower foaf:firstName ?fn .
  ?follower foaf:lastName ?ln .
  OPTIONAL { ?follower lib:gender ?g }
}
```