Thuillier Johann 336104120
Bismuth Samuel 342533064

Assignment 3 – Databases

mongoDB :

In the mongoDB folder you can find a code in Java using Maven Central which insert some data in mongoDB.

Query

```
use admin;

db.Library.insert( {Book_name : "Harry Potter", Author : "JK Rollwing", Editor : "Folio Junior", Edit_year : 2004, Book : "The book himself download from pdf file in java."} );
```

Code in java which insert of the entire book

```
package org.mongodb.mongoDB;

import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.text.PDFTextStripper;
import org.bson.Document;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

public class Insert
{
	@SuppressWarnings("resource")
	public static void main( String[] args )
	{
		MongoClientURI connectionString = new MongoClientURI("mongodb://127.0.0.1:27017");
```

```
                MongoDatabase database = new
MongoClient(connectionString).getDatabase("admin");
                MongoCollection<Document> collection =
database.getCollection("Library");
                Logger mongoLogger = Logger.getLogger("org.mongodb.driver");
                mongoLogger.setLevel(Level.SEVERE);
                try
                {
                String currentFolder = System.getProperty("user.dir");
                PDDocument pdf_book = PDDocument.load(new File(currentFolder + "/
Harry Potter.pdf"));
                String text = "";
                if (!pdf_book.isEncrypted())
                {
                        PDFTextStripper stripper = new PDFTextStripper();
                        text = stripper.getText(pdf_book);
                }
                pdf_book.close();
                Document document = new Document()
                                .append("Book name", "Harry Potter")
                                .append("Author", "JK Rollwing")
                                .append("Editor", "Folio Junior")
                                .append("Edit year", 2004)
                                .append("Book", text);
                collection.insertOne(document);
                }
                catch (IOException ex)
                {
                        System.out.println(ex);
                }
        }
}
```

Map reduce

```
use admin;

map = function() {
   var book_words = this.Book.split(" ");  // split with your logic.
   for (var i = 0; i < book_words.length; i++)
   {
      emit(book_words[i].length, 1);
   }
};

reduce = function(key, value) {
   return value.length;
};
```

```
db.Library.mapReduce(map, reduce, {out:'reducing'});

db.reducing.find();
```

Configuration

please run
mongo < mapReduce.js
from your linux terminal to run the mapReduce.

Neo4J :

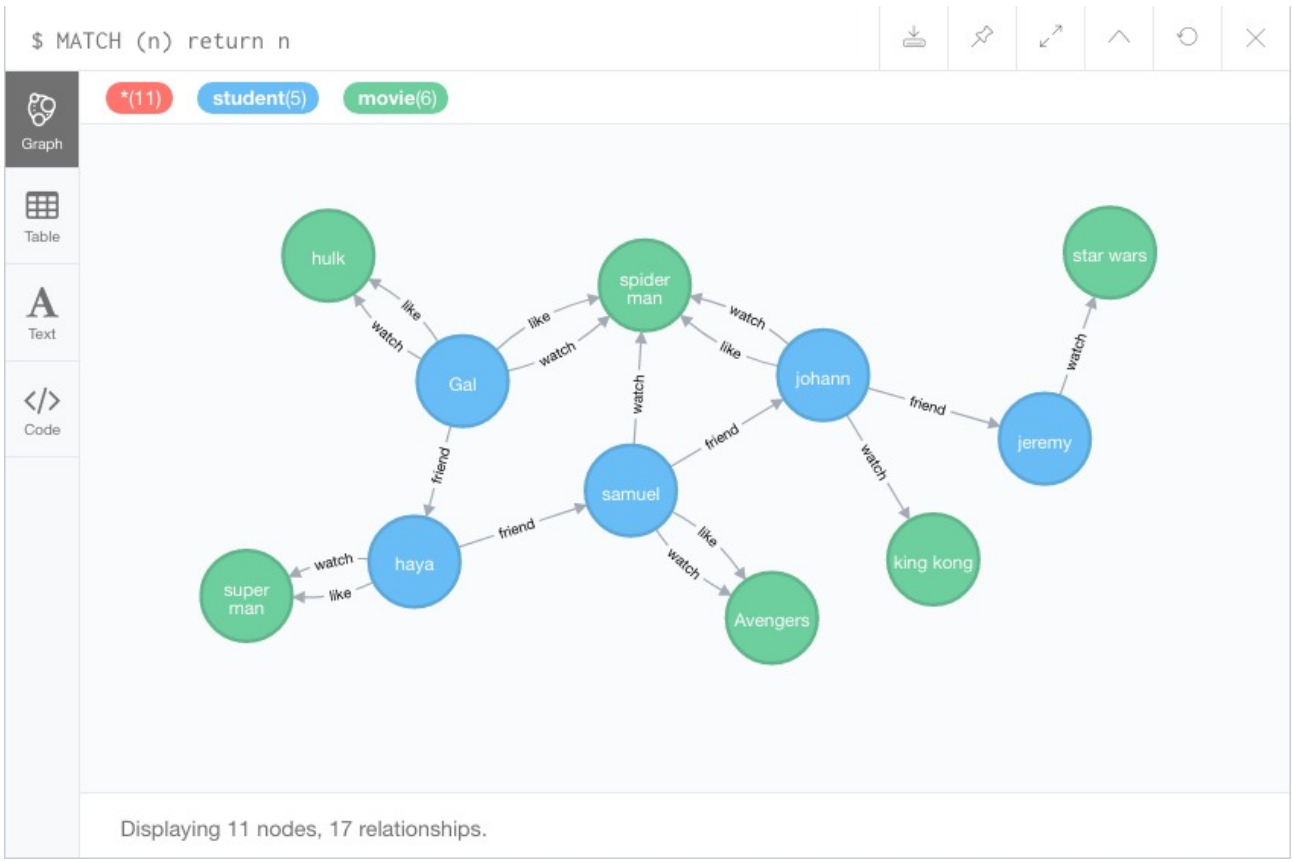
Query in Cypher for neo4j

```
// Match movies that at least two of Gal's friends liked or watched.

MATCH (g:student{ name:'Gal' })-[:friend*1..3]-(s:student)-[:like |:watch]-
>(m:movie)
WITH m, COUNT(DISTINCT s) as num_student
WHERE num_student >= 2
WITH COLLECT(m) as team_movies

// Match movies that Gal watched AND liked that also include in team_movies.

MATCH (movies:movie)<-[:watch]-(gal:student{name:'Gal'})-[:like]-
>(movies:movie)
WHERE (movies IN team_movies)
return movies
```

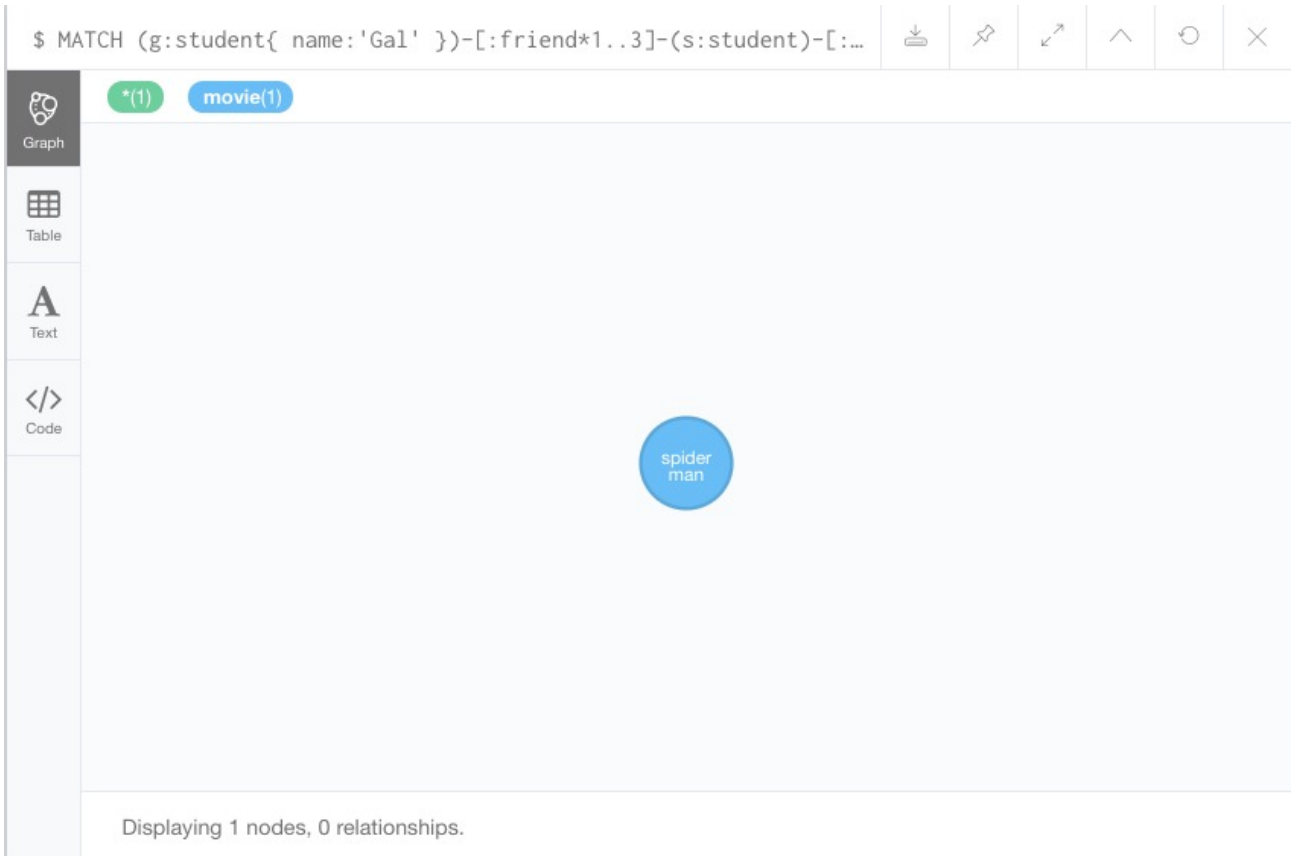
Configuration

To verify our query we used deo4j Desktop.
After creating an account, you may be able to create your own database and run our query.

Screenshots



The database using Neo4j desktop.



The result query.

ElasticSearch :

Insert query

```
POST /books/book/1
{
  "title": "Harry Potter",
  "Author": "JK Rollwing",
  "Genre": "Fantasy",
  "Editor": "Folio Junior",
  "Edite year": 2004,
  "Topic": "You're a wizard Harry."
}
```

Search query

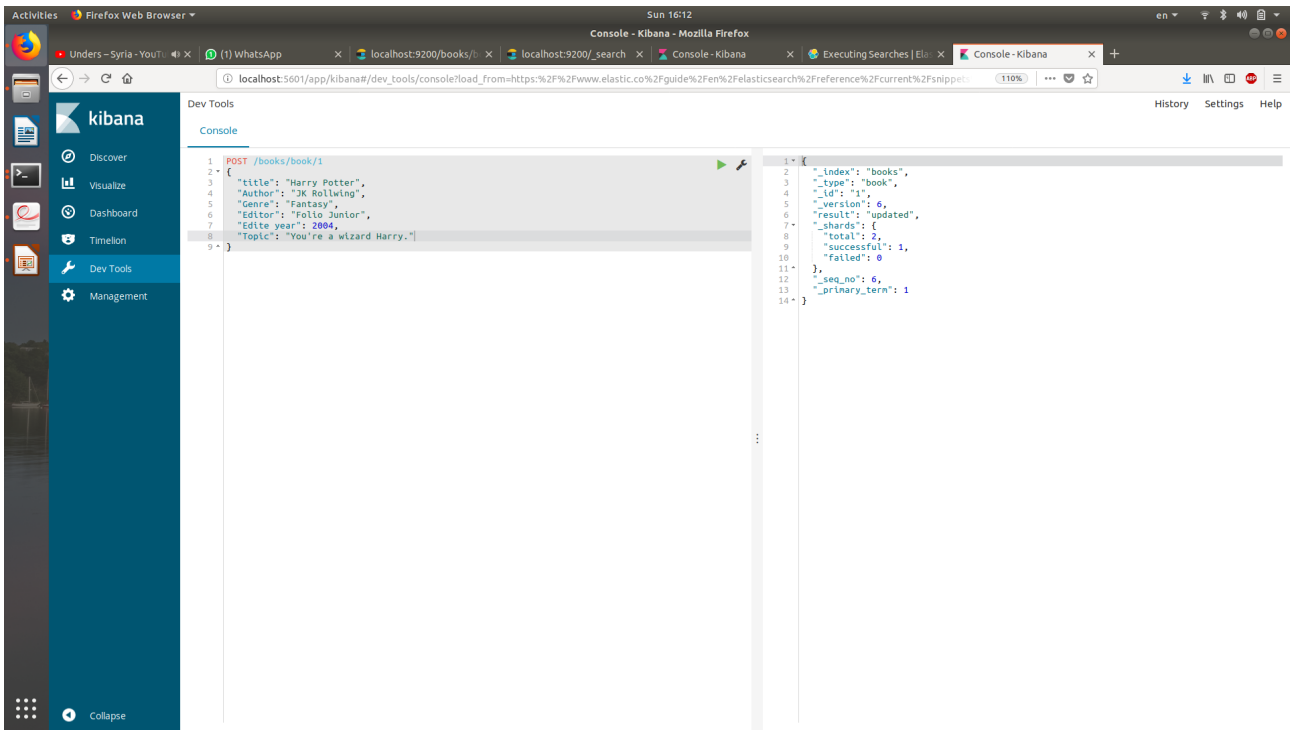
```
GET /books/book/_search
{
  "query": {
    "bool": {
      "must":  { "range": {"Edite year": { "gte": 2000 } } },
    "filter":  [
        { "match": { "Topic": "Science Fiction" } } ,
        { "match": { "Topic": "reality" } } ,
        { "match":  { "Genre": "Science Fiction" } } ]
    }
  }
}
```

Configuration

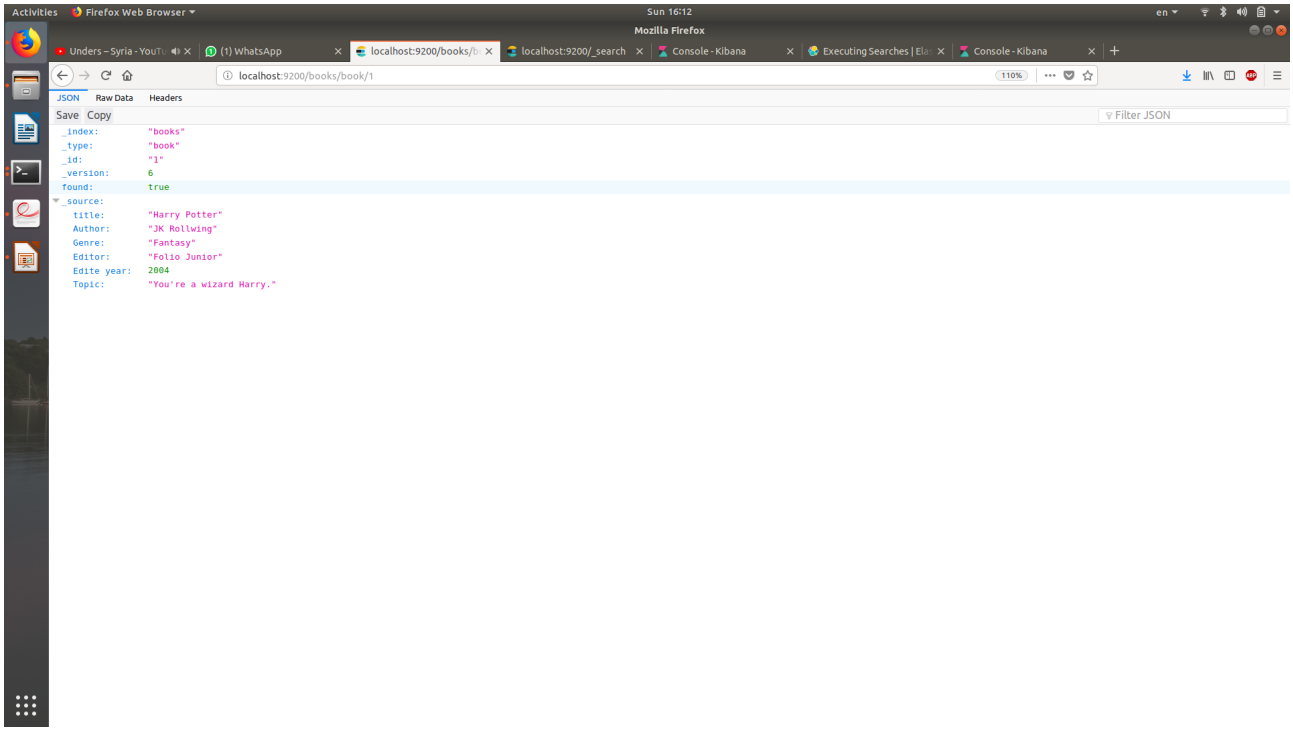
To add the data or seach data you need first to download kibana :
https://www.elastic.co/downloads/kibana
then run ./kibana from thje linux terminal and open the next link in you browser :
http://localhost:5601/app/kibana#/dev_tools/console?_g=()

Then put the query in the console and the data is now added.

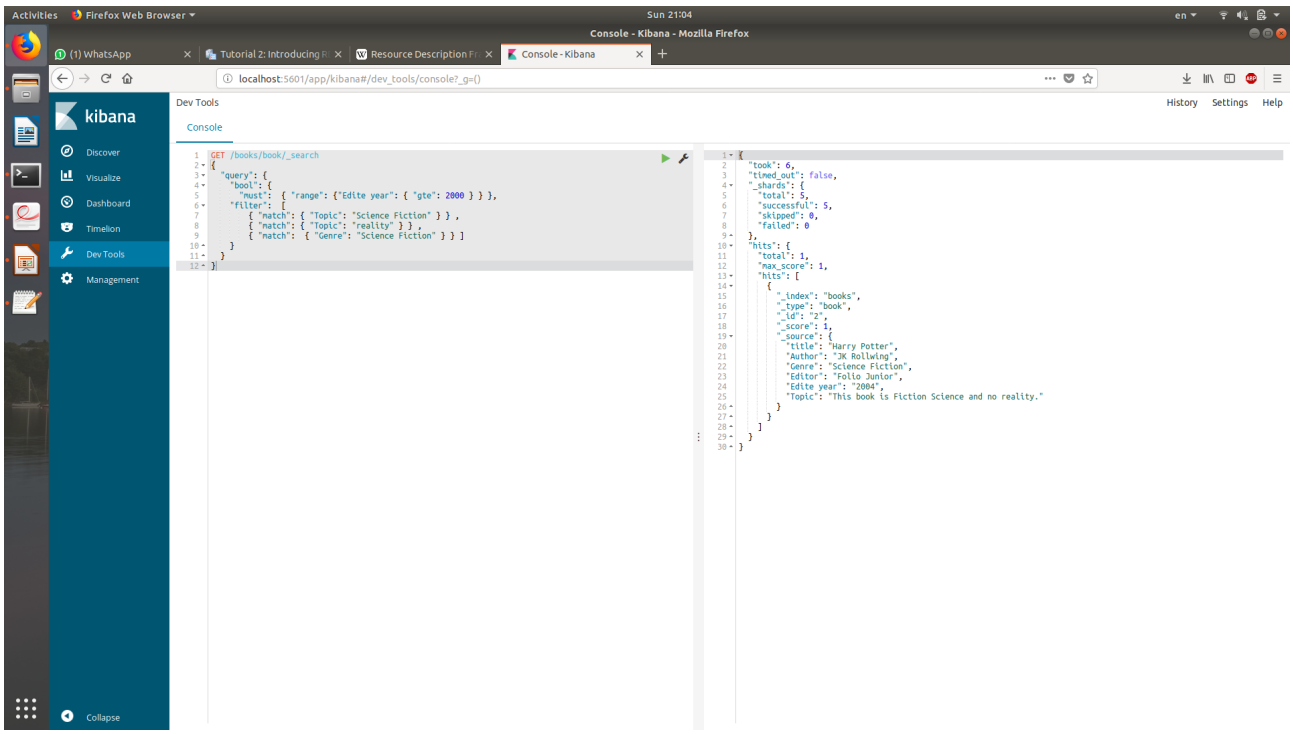
Screenshots



Insert command.



Result of the insert.



Search command and result.

X-path :

Query

```
for $i in //country return $i[sum(city /@num ) > 1000000] /@name
```

Xml code for the test

```
<?xml version="1.0"?>
 <root>
   <country name = "Israel">
     <city num= "1029"> Ariel </city>
     <city num = "79833"> Tel Aviv </city>
     <city num = "89222"> Jerusalem </city>
     <city num = "53728"> Haifa </city>
     <city num = "250"> Kokhav Hachakhar </city>
   </country>
   <country name = "France">
     <city num = "62234"> Lyon </city>
     <city num = "7373663"> Paris </city>
   </country>
   <country name = "Espagne">
     <city num = "39430223"> Madrid </city>
     <city num = "7373663"> Barcelone </city>
   </country>
 </root>
```

Configuration

You can run xpath from this tester :
https://www.freeformatter.com/xpath-tester.html#ad-output
copy the xml file.
Then, compute you xpath code.

Stream :

```
package stream;

import java.util.Arrays;

public class StreamUse {

	public static void main(String[] args) {
		Arrays.asList('a', 'b', 'c', 'g', 'H', 'i', 't', 'u', 'v')
		.stream()
		.forEach(
				(c)
				->
				{
					if (c > 'g' && c < 'o')
						System.out.println((char) ('z' - c + 'a'));
					else if (c > 'G' && c < 'O')
						System.out.println((char) ('Z' - c + 'A'));
				}
				);
	}
}
```

SPARQL & RDF :

RDF table

S	P	O
cv:111	tto:name	ttc:ישראל ישראל
cv:222	tto:name	ttc:פלוני אלמוני
cv:333	tto:name	ttc:גון סמיט
cv:444	tto:name	ttc:ראובן אריאל
cv:111	gb:age	15
cv:222	gb:age	2
cv:333	gb:age	30
cv:444	gb:age	81
cv:111	aws:father	cv:444
cv:222	aws:father	cv:333
cv:333	aws:father	cv:444
cv:444	aws:father	cv:555

SPARQL query

```
SELECT ?name
WHERE
{
   ?person dbp:name ?name .
   ?person aws:father ?father_variable .
   ?father_variable aws:father ?grandFather .
   ?grandFather aws:father ?444 .
}
```

TF-IDF :

Q: איזה יום היום

1: היום שימשי מאוד בחוץ.

2: היום יום חמישי.

3: ! איזה יום נעים היום

4: היום יום הולדת ליונתן.

Q	היום	יום	איזה	#words
1.	1	0	0	4
2.	1	1	0	3
3.	1	1	1	4
4.	1	1	0	4
#Doc	4	3	1	

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

	Tf Idf score
1.	(1/4)*log(4/4) = 0
2.	(1/3)*log(4/4) + (1/3)*log(4/3) = 0.138
3.	(1/4)*log(4/4) + (1/4)*log(4/3) + (1/4)*log(4) = **0.603**
4.	(1/4)*log(4/4) + (1/4)*log(4/3) = 0.103

The third sentence is the most compatible result according to TF IDF.

Source for the formula : http://www.cnblogs.com/youth0826/archive/2012/08/11/2633688.html