Data Bases - assignment 3

elad sezanayev 211909940
Liav Levi 206603193

MongoDB:

1.

```
db.books.insert({"title":"book1",
                 "author": "elad",
                 "Publish": "ariel",
                 "year": "2022",
                 "book":  "SUPPOSING that Truth is a
                           woman--what then?"})
```

2.

```
var mapper = function() {
	var text = this.book.replace(/[^a-zA-Z]/gi, " ");

	text = text.split(" ");

	for (var i = 0; i <= text.length - 1; ++i) {
		if(text[i].length != 0)
			emit(text[i].length, 1);
	}
};

var reduce = function(length, count) { return Array.sum(count);};

db.books.mapReduce(mapper, reduce, {out:"example"})
```

Neo4j:

```
MATCH (p:post)<-[:publish]-(d:person {name:"Dani}) WITH COLLECT(p)
AS dani_posts
MATCH (f:person)<-[:friend*1..3]-(d:person {name:'Dani'})
```

WHERE f.age > d.age AND ALL (p IN dani_posts WHERE (f)-[:like]->(p))
RETURN f

XPath:

country[city/@num > 1000000]/@name

Stream:

Here is the main file (only the code) with the answers:

```
import javafx.util.Pair;
import java.util.LinkedList;
import java.util.List;
import java.util.stream.Collectors;

/**
 * Created by Avigail on 5/20/2017.
 */
public class Main {
    public static void main(String[] args) {
        List<Course> courses = new LinkedList<Course>() {{
            add(new Course(30, "Data Structures", 3.5f));
            add(new Course(32, "Geometry", 6));
            add(new Course(35, "Algebra", 2.5f));
            add(new Course(37, "English", 7));
        }};

        List<Student> students = new LinkedList<Student>() {{
            add(
                    new Student("Moshe", 21, 1, new LinkedList<Grade>() {{
                        add(new Grade(courses.stream().filter(c -> c.getId() == 30).findFirst().get(), 67));
                        add(new Grade(courses.stream().filter(c -> c.getId() == 32).findFirst().get(), 89));
                        add(new Grade(courses.stream().filter(c -> c.getId() == 35).findFirst().get(), 67));
                        add(new Grade(courses.stream().filter(c -> c.getId() == 37).findFirst().get(), 89));
                    }})
            );
            add(
                    new Student("Yossi", 26, 2, new LinkedList<Grade>() {{
                        add(new Grade(courses.stream().filter(c -> c.getId() == 30).findFirst().get(), 100));
                        add(new Grade(courses.stream().filter(c -> c.getId() == 32).findFirst().get(), 67));
                        add(new Grade(courses.stream().filter(c -> c.getId() == 35).findFirst().get(), 89));
                    }})
            );
            add(
                    new Student("Natasha", 30, 3, new LinkedList<Grade>() {{
                        add(new Grade(courses.stream().filter(c -> c.getId() == 30).findFirst().get(), 67));
                        add(new Grade(courses.stream().filter(c -> c.getId() == 37).findFirst().get(), 80));
```

```
            }})
        );
    }};


    /**
     * This is the answer for 1:
     */
    students.forEach(x -> System.out.println(x.getName() + ": " + x
            .getGrades()
            .stream()
            .mapToDouble(Grade::getValue)
            .average()
            .orElse(0.0)));

    /**
     * This is
     * the answer for 2:
     */

    List<Grade> grades = students.stream()
            .flatMap(s -> s
                    .getGrades()
                    .stream())
            .collect(Collectors.toList());
    courses.forEach(c -> System.out.println(c.getName() + " : " +
            grades
                    .stream()
                    .filter(g -> g.getCourse() == c)
                    .mapToDouble(Grade::getValue)
                    .average()
                    .orElse(0.0)));


  }
}
```

Spark & RDF:

1.

Subject	Predicate	Object
123	F_name	Dani
123	L_name	Choen
123	Age	24
123	Father_id	333
333	F_name	Michal
333	L_name	Levi
333	Age	56
333	Father_id	444
444	F_name	Reuven
444	L_name	Levi
444	Age	80
444	Father_id	111

2.

```
SELECT ?grand WHERE {

        ?grand Father_id ?dad .
        ?dad Father_id 444 .
        ?grand Age ?age .
        FILTER(?age > 25) .

}
```

TF - IDF:

Q: what day is today

1: Very sunny day outside.

2: Today is Thursday.

3: What a pleasant day today!

4: Today is Jonathan's birthday.

The rankig of the numbred sentences above by TF-IDF is:
i. What a pleasant day today!
ii. Today is Thursday.
iii. Today is Jonathan's birthday.
iv. Very sunny day outside.

Or in other words: 3->2->4->1 as we wrote at the lecture.