**MONGO-DB:**

1. db.books.find({$or:[{$and:[{"for\_age":{$gte:10}},{"publishing\_year":{$lt:2000}}]},{$and:[{"authors":"Ruth Cohen"},{"rank":{$gte:3}}]}]}).pretty();
2. mapper = function ()

{  
 let words = this.the\_book.split(" ");  
 for (let i = 0; i < words.length; i++) {  
 var key = words[i].length;  
 emit(key, 1);  
 }  
};

reducer = function (key, count)   
 return Array.sum(count);  
 };

db.books.mapReduce(mapper, reducer, {  
 query: { publishing\_year: { $ne: 2000 } },  
 out: "count\_words"}  
 );

**ELASTIC-SEARCH:**

1. curl -XPUT “http://localhost:4000/books” -d “{\”name\”:\”b\”, \”authors\”:\”Matan and Reut\” ,\”Genre\”:\”children\”, \”publisher\”:\”Ariel University\”, \”Year\”:\”2021\”, \”Summary\”:\”The b\”}”
2. curl -XGET “http://localhost:4000/books/\_search ” -d   
   “{\"query\”: {\"bool\":   
   {\"filter\”: [{\"match\": {\"Genre\": \"Science Fiction\"}, {\"match\": {\”Summary\”:{\”query\”:   
    \“reality Fiction Science\”, \“operator\”: \”AND\”}}, {\"range\": {\"year\":{\"gte\" : 2999}}}]  
   }}}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | What | A | Day | Num of words in sentence |
| Sen 1 | 0 | 1 | 1 | 5 |
| Sen 2 | 1 | 0 | 0 | 4 |
| Sen 3 | 1 | 0 | 1 | 7 |
| Sen 4 | 0 | 1 | 1 | 6 |
| Sen 5 | 1 | 1 | 1 | 8 |
|  | 3 | 3 | 4 |  |

**TF-IDF:**

1. 1/5\*log(5/3) + 1/5\*log(5/4) = 0.0637
2. 1/4\*log(5/3) = 0.0554
3. 1/7\*log(5/3) + 1/7\*log(5/4) = 0.0455
4. 1/6\*log(5/3) + 1/6\*log(5/4) = 0.0531
5. 1/8\*log(5/3) + 1/8\*log(5/3) + 1/8\*log(5/4) = 0.0675

סדר הדירוג (הראשון משמאל): 5,1,2,4,3

**SPARQL & RDF:**

|  |  |  |
| --- | --- | --- |
| Object | Predicate | Subject |
| ישראל ישראלי | Name | 111 |
| 15 | Age | 111 |
| 444 | Father\_ID | 111 |
| פלוני אלמוני | Name | 222 |
| 2 | Age | 222 |
| 333 | Father\_ID | 222 |
| ג'ון סמית | Name | 333 |
| 30 | Age | 333 |
| 444 | Father\_ID | 333 |
| ראובן אריאל | Name | 444 |
| 81 | Age | 444 |
| 555 | Father\_ID | 444 |

SELECT ?n WHERE {

?id Father\_ID “444”.

?x Father\_ID ?id.

?x name ?n.

}

**STREAM:**

import java.util.stream.\*;  
  
public class count *{* public static void main(String[] args) *{* String s = "SUPPOSING that Truth is a woman what then";  
 String[] arr = s.split(" ");  
 Stream.*of*(Stream.*of*(arr).filter(x -> x.length() > 4 && x.length() < 10).collect(Collectors.*groupingBy*(count::*setName*, Collectors.*counting*()))).forEach(System.*out*::println);  
 *}* static public String setName(String s)  
 *{* return ""+s.length()+"\_letters";  
 *}  
}*

**X-PATHK:**

//country/@name[sum(../city/@num)>1000000]

**NEO4J:**

MATCH (n)-[:friend\*0..3]-(m)

WHERE n.salary>= 19999 and n.address = "Tel Aviv"

WITH n, collect(m) as nodes

WHERE ALL(x IN nodes WHERE x.salary>= 19999)

WITH n ORDER BY n.name

RETURN collect(DISTINCT n.name)