Bazy danych – NoSQL MongoDB – zadania

> Mateusz Nabywaniec data laboratorium 27.11.19 r. data wykonania 15.12.19 r.

- 1. Wykorzystując bazę danych yelp dataset wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty:
- a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
db.getCollection('business').distinct("city").sort()
```

b. Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (włącznie).

```
db.getCollection('review').count({
    'date': {"$gte" : ("2011-01-01")}
})
```

c. Zwróć dane wszystkich zamkniętych (open) firm (business) z pól: nazwa, adres, gwiazdki (stars).

```
db.getCollection('business').find({
    'open': false
    },{
    'name' : 1,
    'full_address' : 1,
    'stars' : 1
     }
)
```

d. Zwróć dane wszystkich użytkowników (user), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (funny lub useful), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2012. Wynik posortuj alfabetycznie na podstawie liczby (tip).

Na początku tworzę kolekcje tips\_amount zawierającą sumę tipów dla każdej firmy. Następnie odwołuję się do tej tabeli i dopasowuję nazwę biznesu do jego id. Na końcu sortuję wynik najpierw wg liczby napiwków, a potem alfabetycznie.

```
$sum: 1
              }
         }
    },
{
         $out: "tips_amount"
    }
])
db.business.aggregate([{
         $lookup: {
   from: "tips_amount",
              localField : 'business_id',
              foreignField : '_id',
              as : "tips"
         }
    },
{
         $unwind: {
              path : "$tips",
              preserveNullAndEmptyArrays: true
    },
{
         $project: {
    _id : '$_id',
    business_id : "$business_id",
              name : '\$name',
              tips: "$tips.tips_amount"
         }
    },
{
         $sort: {
              "tips" : -1,
"name" : 1
         }
    }
])
```

f. Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji. Wynik ogranicz do recenzji, które uzyskały min 4.0 gwiazdki.

```
db.review.aggregate([{
         $group : {
    _id : "$business_id",
             }
    },
    {
         $out: "avg_stars"
    }
])
db.business.aggregate([{
        $lookup: {
    from: "avg_stars",
             localField : 'business_id',
foreignField : '_id',
             as : "average"
         }
    },
{
         $unwind: {
             path: "$average",
             preserveNullAndEmptyArrays: true
```

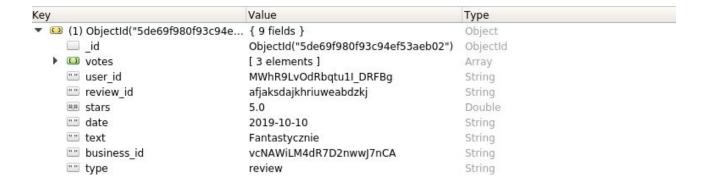
```
}
                             },
{
                                 $project: {
    _id : '$_id',
                                      business_id : "$business_id",
                                      name : '$name',
                                      avg_stars : {$filer :{
    input: "$average",
                                          as : "average"
cond : {
                                               "$average.avg_stars" : {$gte : 4}
                                           }
                                      }
                                  }
                             },
{
                                  $sort: {
                                      "avg_stars" : -1
                             }
                         ])
g. Usuń wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0.
                          db.business.deleteMany({stars : {$eq : 2}})
2. Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej recenzji (review). Wykonaj przykładowe wywołanie.
                         function insertReview(votes, user_id, review_id, stars, date, text,
                         business_id){
                             db.review.insert({
                                  votes:votes,
                                  user_id:user_id,
                                  review_id:review_id,
                                  stars:stars,
                                  date:date,
                                  text:text,
                                  business_id:business_id,
                                  type: 'review'
                             });
                         }
Przykładowe wywołanie:
                         insertReview([0,10,10], "MWhR9LvOdRbqtu1I_DRFBg",
                         "afjaksdajkhriuweabdzkj", Number(5),
```

"2019-10-10", "Fantastycznie", "vcNAWiLM4dR7D2nwwJ7nCA")

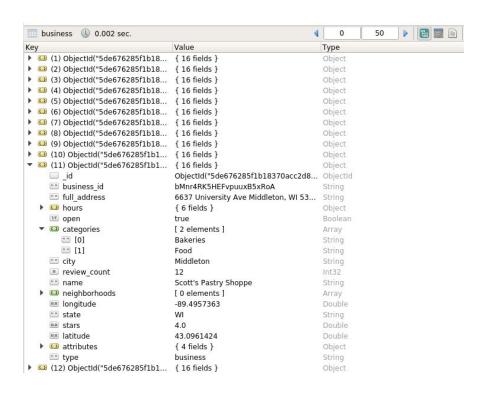
db.review.find({

})

date: "2019-10-10"



3. Zdefiniuj funkcję (MongoDB), która zwróci wszystkie biznesy (business), w których w kategorii znajduje się podana przez użytkownika cechę. Wartość kategorii należy przekazać do funkcji jako parametr. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.



4. Zdefiniuj funkcję (MongoDB), która umożliwi modyfikację nazwy użytkownika (user) na podstawie podanego id. Id oraz nazwa mają być przekazywane jako parametry.

```
function updateUserName(user_id, new_name){
    db.user.update(
          {user_id:user_id},
          {$set: {name: new_name}}
    )
}
```

```
updateUserName("5Xh4Qc3rxhAQ_NcNtxLssQ", "Mateusz Nab")
db.user.find({name:"Mateusz Nab"})
```

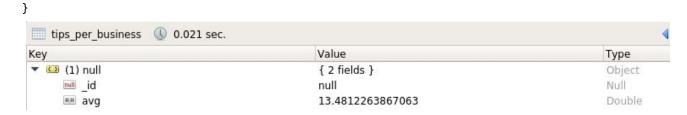
5. Zwróć średnią ilość wszystkich wskazówek/napiwków dla każdego z biznesów, wykorzystaj map reduce.

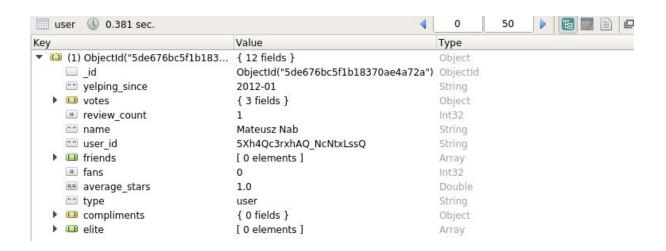
```
var mapFunc = function(){
    emit(this.business_id, 1)
}

var reduceFunc = function (business_id, tips){
    return Array.sum(tips)

db.tip.mapReduce(
    mapFunc,
    reduceFunc,
    {out : "tips_per_business"}
)

db.tips_per_business.aggregate([
    {$group: { _id : null, avg: {$avg : "$value"}}}
])
```





- 6. Odwzoruj wszystkie zadania z punktu 1 w języku programowania (np. JAVA) z pomocą API do MongoDB. Wykorzystaj dla każdego zadania odrębną metodę.
- a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
private List<String> exerciseA(){
    return db.getCollection("business").distinct("city");
}

public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
```

```
List<String> businesses = mongoLab.exerciseA();
                 Collections.sort(businesses);
                 for(String business : businesses) {
                          System.out.println(business);
                 //mongoLab.showCollections();
        }
b. Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (włącznie).
        private long exerciseB(){
                 DBObject query = new BasicDBObject("date", new BasicDBObject("$gte", "2011-01-01"));
                 return db.getCollection("review").count(guery);
        }
c. Zwróć dane wszystkich zamknietych (open) firm (business) z pól: nazwa, adres, gwiazdki (stars).
        private void exerciseC(){
                 DBCursor cursor = db.getCollection("business")
                                   .find(
                                           new BasicDBObject("open", false),
                                                    new BasicDBObject().append("name",1)
                                                             .append("full_address",1)
                                                             .append("stars", 1));
                 while (cursor.hasNext()) {
                          System.out.println(cursor.next());
                 }
        }
d. Zwróć dane wszystkich użytkowników (user), którzy nie uzyskali ani jednego
pozytywnego głosu z kategorii (funny lub useful), wynik posortuj alfabetycznie na podstawie imienia użytkownika.
        private void exerciseD() {
                 DBObject clauseFunny = new BasicDBObject("votes.funny", new BasicDBObject("$lte", 0));
                 DBObject clauseUseful = new BasicDBObject("votes.useful", new BasicDBObject("$lte", 0));
                 BasicDBList and = new BasicDBList();
                 and.add(clauseFunny);
                 and.add(clauseUseful);
                 DBObject guery = new BasicDBObject("$and", and);
                 DBCursor cursor = db.getCollection("user").
                                   find(query).sort(new BasicDBObject("name",1));
                 while (cursor.hasNext()) {
                          System.out.println(cursor.next());
                 }
        }
e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2012.
Wynik posortuj alfabetycznie na podstawie liczby (tip).
        private AggregateIterable<Document> exerciseE(){
                 AggregateIterable<Document> tipsAmount =
database.getCollection("tips").aggregate(Arrays.asList(
                                   new Document("$match", new Document("date",
Pattern.compile("2012"))),
                                   new Document("$group", new Document("_id", "$business_id")
                                                    .append("tips_amount", new Document("$sum", 1))),
```

```
new Document("$out", "tips amount")
                                  ));
                 AggregateIterable<Document> result =
database.getCollection("business").aggregate(Arrays.asList(
                                  new Document("$lookup", new Document("from", "tips_amount")
                                                    .append("localField", "business_id")
                                                    append("foreignField", "_id")
                                                    .append("as", "tips")),
                                  new Document("$unwind", new Document("path", "$tips")
                                                    .append("preserveNullAndEmptyArrays", true)),
                                  new Document("$project",new Document("_id","$_id")
                                              .append("business id", "$business id")
                                              .append("name","$name")
                                              .append("tips", "$tips.tips_amount")),
                                  new Document("$sort", new Document("tips",-1))
                          ));
                 return result:
        }
f. Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji. Wynik
ogranicz do recenzji, które uzyskały min 4.0 gwiazdki.
private AggregateIterable<Document> exerciseF(){
                 AggregateIterable<Document> doc1 = database.getCollection("review").aggregate(Arrays.asList(
                                  new Document("group", new Document("_id", "$business_id")
                                                    .append("avg_stars", new Document("$avg", "$stars"))),
                                  new Document("$out", "avg stars")));
                 AggregateIterable<Document> result = database.getCollection("business").aggregate(Arrays.asList(
                                  new Document("$lookup", new Document("from", "avg stars")
                                                    .append("localField", "business_id")
                                                    .append("foreignField"," id")
                                                    .append("as", "averages")),
                                  new Document("$unwind", new Document("path", "$averages")
                                                    .append("preserveNullAndEmptyArrays", true)),
                                  new Document("$project", new Document("_id","$_id")
                                              .append("business id", "$business id")
                                              .append("name","$name")
                                              .append("avg_stars",
                                                             new Document("$filter", new Document("input",
"$averages")
                                                                              .append("as", "average")
                                                                              .append("cond", new
Document("avg stars", new Document("$gte", (new Document("averages.avg stars", 4))))))),
                                  new Document("$sort", new Document("name", 1))
                                                    ));
                                  return result:
        }
g. Usuń wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0.
        private void exerciseG()
                 database.getCollection("business").
                          deleteMany(new Document("stars", new Document("$eq",2.0)));
        }
```

7. Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: klientów, zakupu oraz przedmiotu zakupu. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych.

```
db.customer.insert({
    first_name: "Mateusz",
    last name: "Kowalski",
    birthdate: ISODate("1998-01-30"),
    address: {
        street: "Mickiewicza 10",
        city: "Krakow"
    },
    phone: ["512124125", "623231352"]
    mail: "mkowalski@mail.eu"
})
db.product.insert({
    name : "Maka Lubelska",
price : 4.5,
category : "Food",
    units_in_stock : 1000,
    expiry_date : ISODate("2020-02-11")
})
db.product.insert({
    product_id : "makaron11",
    name: "Makaron",
    price: 6,
    category : "Food",
    units_in_stock : 500,
    expiry date : ISODate("2020-03-01")
})
db.purchase.insert({
    purchase_id : "mkowalpuch1",
    client_id : "mkowalski1mick10krak1",
    products_list : [{
        product_id : "makalub1",
        units: 10
       },
        product_id : "makaron11",
        units: 5
       }],
    price : 75,
    purchase_date : ISODate("2019-12-11")
})
db.product.insert({
    product_id : "makaron11",
    name: "Makaron",
    price: 6,
    category : "Food",
    units_in_stock : 500,
```

expiry\_data : ISODate("2020-03-01")
})

