**Bazy danych – NoSQL MongoDB – zadania**

**Aleksandra Mazur**

**Tydzień A**

**Wtorek 9:35**

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')**


```
/* 1 */
[
    "Ahwatukee",
    "Anthem",
    "Apache Junction",
    "Arcadia",
    "Atlanta",
    "Avondale",
    "Black Canyon City",
    "Bonnyrigg",
    "Boulder City",
    "Buckeye",
    "C Las Vegas",
    "Cambridge",
    "Carefree",
    "Casa Grande",
    "Cave Creek",
    "Centennial Hills",
    "Central City Village",
    "Central Henderson",
    "Chandler",
    "Chandler-Gilbert",
    "City of Edinburgh",
    "Clark County",
    "Columbus",
    "Coolidge",
    "Cottage Grove",
    "Cramond",
    "Dalkeith",
    "Dane",
    "De Forest",
    "DeForest",
```

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

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

```
880318
```

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

    } )
```


```
/* 1 */
{
    "_id" : ObjectId("5e791b6832bd2e12dd9c6b93"),
    "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
    "name" : "Charter Communications",
    "stars" : 1.5
}

/* 2 */
{
    "_id" : ObjectId("5e791b6832bd2e12dd9c6b9e"),
    "full_address" : "6401 University Ave\nMiddleton, WI 53562",
    "name" : "Crandalls Carryout & Catering",
    "stars" : 4.0
}

/* 3 */
{
    "_id" : ObjectId("5e791b6832bd2e12dd9c6ba2"),
    "full_address" : "6230 University Ave\nMiddleton, WI 53562",
    "name" : "Mi Cocina",
    "stars" : 3.0
}
```

d) Zwróć dane wszystkich użytkowników (user), którzy nie uzyskali jednego pozytywnego głosu z kategorii (funny lub useful), wynik posortuj alfabetycznie według nazwy użytkownika

```
db.getCollection('user')
.find({
    $or:
    [
            {'votes.funny': 0},
            {'votes.useful': 0}
    ]
    }) .sort({'name':1})
```


```
/* 1 */
{
    "_id" : ObjectId("5e791bc82fb64db814984347"),
    "yelping_since" : "2009-08",
    "votes" : {
        "funny" : 0,
        "useful" : 0,
        "cool" : 0
    },
    "review_count" : 1,
    "name" : " Bernard",
    "user_id" : "xP3SPgfgW2vc5Zj5uV8SEA",
    "friends" : [],
    "fans" : 0,
    "average_stars" : 5.0,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}

/* 2 */
{
    "_id" : ObjectId("5e791bcb2fb64db81499416d"),
    "yelping_since" : "2011-12",
    "votes" : {
        "funny" : 0,
        "useful" : 2,
        "cool" : 2
    },
    "review_count" : 10,
    "name" : "'Anastacia",
    "user_id" : "qJLc0rYytqzeVBiTPFtfSA",
    "friends" : [],
    "fans" : 0,
    "average_stars" : 4.5,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}
```


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

```
db.getCollection("tip").aggregate([

    {$match: {"date": /2012/}},

    {$group: {_id: "$business_id", count:{$sum:1}}},

    {$sort:{"count":1}}])
```

```
/* 1 */
{
    "_id" : "edC8euBdO5hxRHRs3RIvZg",
    "count" : 1.0
}

/* 2 */
{
    "_id" : "6EJGBWDGwZb64x5z7XTxmA",
    "count" : 1.0
}

/* 3 */
{
    "_id" : "OrXzZB0M0DbPXx3F_X6GkA",
    "count" : 1.0
}

/* 4 */
{
    "_id" : "a5F7elXHk4gFdTMmDpwZKg",
    "count" : 1.0
}
```

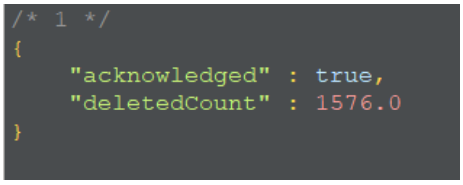

f) Wyznacz, jaką średnią ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji. Wynik ogranicz do recenzji, które uzyskały min 4.0 gwiazdki.

```
db.getCollection("review").aggregate([

    {$group: {_id: "$business_id", avgStars: {$avg: "$stars"}}},

    {$match: {"avgStars": {$gte : 4}}}

])
```


```
/* 1 */
{
    "_id" : "WEfVH5AM2eDpYrj0EVc5lA",
    "avgStars" : 5.0
}

/* 2 */
{
    "_id" : "UDChQLVuz9SYgyyIoEqDkQ",
    "avgStars" : 4.53333333333333
}

/* 3 */
{
    "_id" : "BUHX9Mvp3E3udqNmdlasTg",
    "avgStars" : 4.42857142857143
}

/* 4 */
{
    "_id" : "LUFA1zyso9upnRQnO2qbQA",
    "avgStars" : 4.375
}

/* 5 */
{
    "_id" : "D--QJ1Qtk5B82i_OrcgSPA",
    "avgStars" : 4.0
}
```

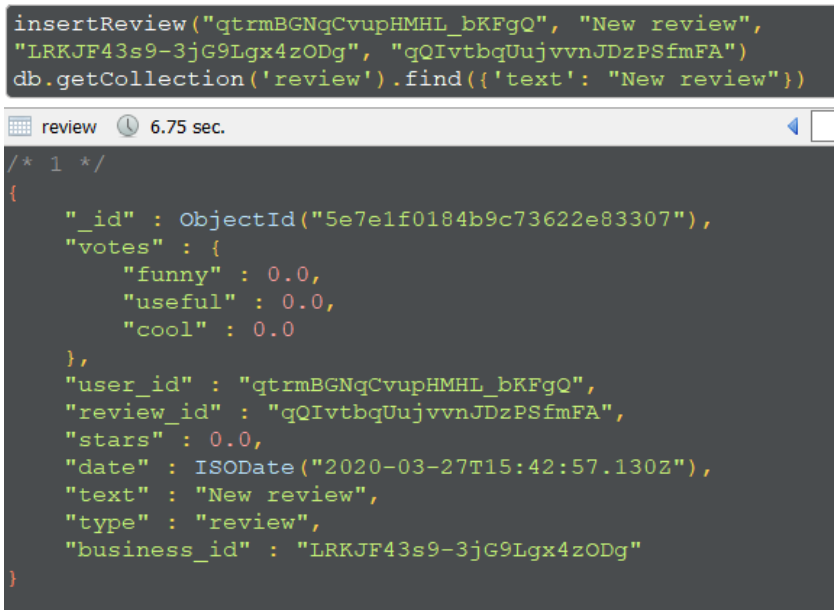
g) Usuń wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0.

```
db.getCollection('business').deleteMany({"stars": 2.0})
```




2. Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej recenzji (review). Wykonaj przykładowe wywołanie.

```
function insertReview(user_id, text, business_id, review_id)
{
    db.review.insert({
        votes: {
            funny: 0,
            useful: 0,
            cool: 0
            },
        user_id: user_id,
        review_id: review_id,
        stars: 0,
        date: new Date(),
        text: text,
        type: "review",
        business_id: business_id
    })
}

insertReview("qtrmBGNqCvupHMHL_bKFgQ", "New review", "LRKJF43s9-3jG9Lgx4zODg",
"qQIvtbqUujvvnJDzPSfmFA")

db.getCollection('review').find({'text': "New review"})
```



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

```
function getBusinessWithCategory(category){
    return db.getCollection('business').find({"categories": category})
}
getBusinessWithCategory("Food")
```

```
/* 1 */
{
    "_id" : ObjectId("5e791b6832bd2e12dd9c6b8f"),
    "business_id" : "LRKJF43s9-3jG9Lgx4zODg",
    "full_address" : "4910 County Rd V\nDe Forest, WI 53532",
    "hours" : {
        "Monday" : {
            "close" : "22:00",
            "open" : "10:30"
        },
        "Tuesday" : {
            "close" : "22:00",
            "open" : "10:30"
        },
        "Friday" : {
            "close" : "22:00",
            "open" : "10:30"
        },
        "Wednesday" : {
            "close" : "22:00",
            "open" : "10:30"
        },
        "Thursday" : {
            "close" : "22:00",
            "open" : "10:30"
        },
        "Sunday" : {
            "close" : "22:00",
            "open" : "10:30"
        },
        "Saturday" : {
            "close" : "22:00",
            "open" : "10:30"
        }
    },
```

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

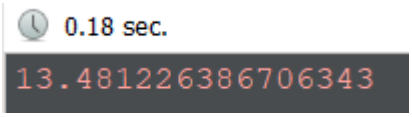
```
function changeUserName(user_id, newName)
{
    db.user.update(
    {user_id: user_id},
    {$set: {name: newName}}
    )
}
changeUserName("qtrmBGNqCvupHMHL_bKFgQ", "newName")
db.getCollection("user").find({})
```

```
/* 1 */
{
    "_id" : ObjectId("5e791bc62fb64db81497832e"),
    "yelping_since" : "2012-02",
    "votes" : {
        "funny" : 1,
        "useful" : 5,
        "cool" : 0
    },
    "review_count" : 6,
    "name" : "newName",
    "user_id" : "qtrmBGNqCvupHMHL_bKFgQ",
    "friends" : [],
    "fans" : 0,
    "average_stars" : 3.83,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}
```


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

```
var mapFunction = function(){
    var key = this.business_id;
    var value = 1
    emit(key, value);

};

var reduceFunction = function(key, values){
    return Array.sum(values);
}

db.tip.mapReduce(mapFunction, reduceFunction, {out: "count_tip"})

var avg = function(tips) {
    sum = 0;
    for (i = 0; i < tips.length; i++){
        sum += tips[i].value;
    }
    return sum/tips.length;
}

db.count_tip.find()
avg(db.count_tip.find().toArray());
```

Za pomocą map reduce obliczono ilość wskazówek dla każdego z biznesu, a za pomocą dodatkowej funkcji obliczono średnią z tych wartości.




```
/* 1 */
{
    "_id" : "--1emggGHgoG6ipd_RMb-g",
    "value" : 6.0
}

/* 2 */
{
    "_id" : "--5jkZ3-nUPZxUvtcbr8Uw",
    "value" : 16.0
}

/* 3 */
{
    "_id" : "--BlvDO_RG2yElKu9XA1_g",
    "value" : 21.0
}

/* 4 */
{
    "_id" : "--Dl2rW_xO8GuYBomlg9zw",
    "value" : 2.0
}

/* 5 */
{
    "_id" : "--Y_21DOtVDioX5bwF6GIw",
    "value" : 5.0
}

/* 6 */
{
    "_id" : "--jFT2mywe7StuZ2hEjxyA",
    "value" : 3.0
}
```

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.

```
public List<String> getCities(){
          return db.getCollection("business")
                 .distinct("city", String.class)
                 .into(new ArrayList<>());
   }
```

```
public List<String> getCities(){
    return db.getCollection( s: "business")
            .distinct( s: "city", String.class)
            .into(new ArrayList<>());
}

public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    List<String> cities = mongoLab.getCities();
    cities.forEach(System.out::println);

}
```

```
Ahwatukee
Anthem
Apache Junction
Arcadia
Atlanta
Avondale
Black Canyon City
Bonnyrigg
Boulder City
Buckeye
C Las Vegas
Cambridge
Carefree
Casa Grande
Cave Creek
Centennial Hills
Central City Village
Central Henderson
Chandler
Chandler-Gilbert
City of Edinburgh
Clark County
Columbus
Coolidge
Cottage Grove
Cramond
```

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

```
public long getReviewsAfter2011(){
    Document query = new Document("date", new Document("$gte", "2011-01-01"));
    long result = db.getCollection("review").countDocuments(query);
    return result;
}
```

```
public long getReviewsAfter2011(){
    Document query = new Document("date", new Document("$gte", "2011-01-01"));
    long result = db.getCollection( s: "review").countDocuments(query);
    return result;
}

public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    long reviews = mongoLab.getReviewsAfter2011();
    System.out.println(reviews);

}
```

```
880318
```

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

```
public List<Document> getClosedCompanies(){
        return db.getCollection("business")
                        .find(eq("open", false))
                        .projection(fields(include("name", "stars", "fill_address"),
                                excludeId()))
                        .into(new ArrayList<>());
}
```

```
public List<Document> getClosedCompanies(){
    return db.getCollection( s: "business") MongoCollection<Document>
                .find(eq( fieldName: "open", value: false)) FindIterable<Document>
                .projection(fields(include( ...fieldNames: "name", "stars", "fill_address"),
                        excludeId())) FindIterable<Document>
                .into(new ArrayList<>());
    }
public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    List <Document> companies = mongoLab.getClosedCompanies();
    companies.forEach(System.out::println);
}
```

```
Document{{name=Charter Communications, stars=1.5}}
Document{{name=Crandalls Carryout & Catering, stars=4.0}}
Document{{name=Mi Cocina, stars=3.0}}
Document{{name=Soup Factory, stars=3.0}}
Document{{name=Deli Roma, stars=4.0}}
Document{{name=Java Detour, stars=4.5}}
Document{{name=Bongo Video, stars=5.0}}
Document{{name=Rocky Rococo Pan Style Pizza, stars=1.5}}
Document{{name=Designs By the Bay, stars=2.5}}
Document{{name=Williamson Bikes & Fitness, stars=3.5}}
Document{{name=Area 51 Vintage Interiors, stars=3.0}}
Document{{name=Dorn True Value Hardware, stars=2.5}}
Document{{name=Wong's Garden, stars=3.5}}
Document{{name=Dimitri's Gyros, stars=3.5}}
Document{{name=Poppa Coronofoulos Gyros & Chicago Style Deli, stars=4.0}}
Document{{name=Rossi's, stars=4.0}}
Document{{name=Groff's Buckeye Barber Shop, stars=5.0}}
Document{{name=Packer Inn, stars=3.0}}
Document{{name=American TV & Appliance, stars=1.0}}
```

d) Zwróć dane wszystkich użytkowników (user), którzy nie uzyskali jednego pozytywnego głosu z kategorii (funny lub useful), wynik posortuj alfabetycznie według nazwy użytkownika

```
public List <Document> getUsers(){
        return db.getCollection("user")
                .find(or(eq("votes.funny", 0), eq("votes.useful", 0)))
                .sort(ascending("name"))
                .into(new ArrayList<>());
}
```

```
public List <Document> getUsers(){
    return db.getCollection( s: "user") MongoCollection<Document>
            .find(or(eq( fieldName: "votes.funny", value: 0), eq( fieldName: "votes.useful", value: 0))) FindIterable<Document>
            .sort(ascending( ...fieldNames: "name")) FindIterable<Document>
            .into(new ArrayList<>());
}

public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    List <Document> users = mongoLab.getUsers();
    users.forEach(System.out::println);
}
```

```
Document{{_id=5e791bcf2fb64db8149a985c, yelping_since=2010-10, votes=Document{{funny=0, useful=3, cool=0}}, review_count=3, name=We
Document{{_id=5e791bcf2fb64db8149a9ddf, yelping_since=2012-09, votes=Document{{funny=0, useful=0, cool=0}}, review_count=3, name=We
Document{{_id=5e791bcf2fb64db8149a9fc5, yelping_since=2010-08, votes=Document{{funny=0, useful=1, cool=0}}, review_count=11, name=W
Document{{_id=5e791bcf2fb64db8149aad2a, yelping_since=2012-03, votes=Document{{funny=0, useful=3, cool=0}}, review_count=13, name=W
Document{{_id=5e791bcf2fb64db8149aad3c, yelping_since=2009-12, votes=Document{{funny=0, useful=0, cool=0}}, review_count=2, name=We
Document{{_id=5e791bcf2fb64db8149aaf49, yelping_since=2012-04, votes=Document{{funny=0, useful=5, cool=4}}, review_count=8, name=We
Document{{_id=5e791bcf2fb64db8149aba2b, yelping_since=2014-03, votes=Document{{funny=0, useful=1, cool=0}}, review_count=1, name=We
Document{{_id=5e791bcf2fb64db8149ababf, yelping_since=2012-06, votes=Document{{funny=0, useful=12, cool=7}}, review_count=7, name=W
Document{{_id=5e791bcf2fb64db8149acae3, yelping_since=2011-06, votes=Document{{funny=0, useful=2, cool=0}}, review_count=1, name=We
Document{{_id=5e791bcf2fb64db8149adf49, yelping_since=2013-03, votes=Document{{funny=0, useful=0, cool=0}}, review_count=1, name=We
Document{{_id=5e791bcf2fb64db8149adf9a, yelping_since=2012-12, votes=Document{{funny=0, useful=0, cool=0}}, review_count=1, name=We
Document{{_id=5e791bd02fb64db8149af065, yelping_since=2013-11, votes=Document{{funny=0, useful=0, cool=0}}, review_count=2, name=We
Document{{_id=5e791bd02fb64db8149af9eb, yelping_since=2012-04, votes=Document{{funny=0, useful=8, cool=2}}, review_count=9, name=We
Document{{_id=5e791bd02fb64db8149afafc, yelping_since=2012-11, votes=Document{{funny=0, useful=1, cool=0}}, review_count=1, name=We
Document{{_id=5e791bd02fb64db8149b00e9, yelping_since=2011-08, votes=Document{{funny=0, useful=0, cool=0}}, review_count=4, name=We
Document{{_id=5e791bd02fb64db8149b0899, yelping_since=2013-06, votes=Document{{funny=0, useful=4, cool=1}}, review_count=8, name=We
```

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

```
public List<Document> getTips(){
        return db.getCollection("tip").aggregate(Arrays.asList(
                Aggregates.group("$business_id", Accumulators.sum("count", 1))
        )).into(new ArrayList<>());
}
```

```
public List<Document> getTips(){
    return db.getCollection( s: "tip").aggregate(Arrays.asList(
            Aggregates.group( id: "$business_id", Accumulators.sum( fieldName: "count", expression: 1))
    )).into(new ArrayList<>());
}
public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    List <Document> tips = mongoLab.getTips();
    tips.forEach(System.out::println);
}
```

```
Document{{_id=g_frA48RM0yOF-j7W1NSHQ, count=1}}
Document{{_id=0Ei4jJCiLj5qJQdiWOZSHg, count=2}}
Document{{_id=IAaKCOkZ4FVfyct4x-tUsQ, count=3}}
Document{{_id=O6qz8lLgqsD8sKTatGDxpg, count=3}}
Document{{_id=vD2Z9HLTFzRT0ZOWjJWkdg, count=1}}
Document{{_id=xYjlf3pudcIn7gX9p6idVg, count=2}}
Document{{_id=h8sRoUSjulDu9MfeUTEu6Q, count=1}}
Document{{_id=0vb_YDCtZo0Yk9zHF7rzSA, count=3}}
Document{{_id=fKwCjTBOmwcbl4XtYRZToQ, count=3}}
Document{{_id=sYROE70SrIpz9IsGsGL2Fg, count=9}}
Document{{_id=m5f8H9Pn2AT2624fRadLFA, count=1}}
Document{{_id=dmyytmVobrPfPO6IrKbjuQ, count=1}}
Document{{_id=PD1R_FR9RFNGuLJAXSr2xg, count=6}}
Document{{_id=hrnvVr3hmb2oD3oUnJrNSw, count=29}}
Document{{_id=1OO3fy6z5lxlRuHnnhPStg, count=8}}
Document{{_id=EdPa3d82Dq_za2Wf7Dtysw, count=7}}
Document{{_id=ak6etTfTtO_DLp5BRfd6uQ, count=1}}
Document{{_id=sZnz926GsnJN1TFCg0j16A, count=4}}
```

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

```
public List<Document> getAVGStars(){
    return db.getCollection("review").aggregate(Arrays.asList(
        Aggregates.group("$business_id", Accumulators.avg("avgStars", "$stars")),
        Aggregates.match(Filters.gte("avgStars", 4.0))
    )).into(new ArrayList<>());
}
```

```
public List<Document> getAVGStars(){
    return db.getCollection( s: "review").aggregate(Arrays.asList(
            Aggregates.group( id: "$business_id", Accumulators.avg( fieldName: "avgStars",  expression: "$stars")),
            Aggregates.match(Filters.gte( fieldName: "avgStars",  value: 4.0))
    )).into(new ArrayList<>());
}
public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    List <Document> stars = mongoLab.getAVGStars();
    stars.forEach(System.out::println);
}
```

```
Document{{_id=vv1uvJRBUqi707TA5GuhCg, avgStars=4.3924050632911396}}
Document{{_id=P6Pd2jqw3kd5W9-2zTH53A, avgStars=4.818181818181818}}
Document{{_id=C9xMK4oGiAOVn6lZQ4yOUw, avgStars=5.0}}
Document{{_id=NxyPHhZRSwPFYr1HZCLG0w, avgStars=4.090909090909091}}
Document{{_id=IyQ78mX1JGWx5fM9g8jKeA, avgStars=4.0}}
Document{{_id=brh2UFHRO0bFfq7KUTbQFA, avgStars=4.318181818181818}}
Document{{_id=7hygeQ-R2TcYfkDBYss-_g, avgStars=4.454545454545454}}
Document{{_id=7p0OI2Ld46-EzJkRHGQEJQ, avgStars=4.507936507936508}}
Document{{_id=WzVvYnJZHvgA0Z-weEJztw, avgStars=4.6}}
Document{{_id=PO6gVnfmA-877zuw8n0auw, avgStars=5.0}}
Document{{_id=QZM5WZIu6HxTu0e9rAcDHA, avgStars=4.428571428571429}}
Document{{_id=mFBBtUrWMqikBS-Jn-RiIw, avgStars=4.5675675675675675}}
Document{{_id=Ohcc9-0otBxqD2_pnZEapw, avgStars=4.8}}
Document{{_id=JnWjER-iSptIYSRgoJs50A, avgStars=5.0}}
Document{{_id=zG_wv69bsllw_PWhOmoAKQ, avgStars=4.312925170068027}}
Document{{_id=6SR684hbkq1RvhEIkklqqQ, avgStars=4.0}}
Document{{_id=mBhznLO0cYd3DaIIQUAAxA, avgStars=4.0}}
Document{{_id=BluXDgvhSEMmvyjTiegWoQ, avgStars=4.222222222222222}}
Document{{_id=Fitb1wRtqWsWh6dWBmXqog, avgStars=4.5}}
Document{{_id=KFJ1jBfFkRfyn3AoAUl3YQ, avgStars=4.2}}
Document{{_id=XI605V-ZprXEMhFzGkpkaQ, avgStars=4.0}}
Document{{_id=3_dKevejfGY3eRj2xprRIQ, avgStars=4.382352941176471}}
Document{{_id=-r5aq7vjN6lMFDZ4XYErkQ, avgStars=4.153846153846154}}
Document{{_id=bV9m2RqRuM8BJ3ZYLjoNDQ, avgStars=5.0}}
```

g) Usuń wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0.

```
public void deleteCompaniesWithNote2(){
        db.getCollection("business").deleteMany(eq("stars", 2.0));
}
```

```
public void deleteCompaniesWithNote2(){
    db.getCollection( s: "business").deleteMany(eq( fieldName: "stars",  value: 2.0));
}
public static void main(String[] args) throws UnknownHostException {
    MongoLab mongoLab = new MongoLab();
    mongoLab.deleteCompaniesWithNote2();
}
```

```
db.getCollection('business').find({"stars": 2.0})
```

0.032 sec.

```
Fetched 0 record(s) in 32ms
```

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. Uzasadnij swoją propozycję.

   Baza danych będzie składała się z trzech kolekcji (Clients, Shoppings, Products). Klient będzie posiadał listę zakupów (paragonów), a każde zakupy (jeden paragon) - listę produktów. Poniżej dodano dane do wymienionych wcześniej kolekcji.

a) Klienci

```
function addClient(name, surname, phone, email){
    db.Clients.insert({
        Name: name,
        Surname: surname,
        Phone: phone,
        Email: email,
        Shoppings: []
    })
}

addClient("Jan", "Nowak", 606522626, "jnowak@poczta.pl");
addClient("Adam", "Kowal", 655322123, "akowal@poczta.pl");
addClient("Kinga", "Adamczyk", 678987654, "kadamczyk@poczta.pl");
addClient("Anna", "Baran", 654321987, "abaran@poczta.pl");
addClient("Karol", "Mak", 666543342, "kmak@poczta.pl");
```


```
{
    "_id" : ObjectId("5e81e1317f4287f25112cd34"),
    "Name" : "Jan",
    "Surname" : "Nowak",
    "Phone" : 606522626.0,
    "Email" : "jnowak@poczta.pl",
    "Shoppings" : []
}

/* 2 */
{
    "_id" : ObjectId("5e81e1317f4287f25112cd35"),
    "Name" : "Adam",
    "Surname" : "Kowal",
    "Phone" : 655322123.0,
    "Email" : "akowal@poczta.pl",
    "Shoppings" : []
}

/* 3 */
{
    "_id" : ObjectId("5e81e1317f4287f25112cd36"),
    "Name" : "Kinga",
    "Surname" : "Adamczyk",
    "Phone" : 678987654.0,
    "Email" : "kadamczyk@poczta.pl",
    "Shoppings" : []
}

/* 4 */
{
    "_id" : ObjectId("5e81e1317f4287f25112cd37"),
    "Name" : "Anna",
    "Surname" : "Baran",
    "Phone" : 654321987.0,
    "Email" : "abaran@poczta.pl",
    "Shoppings" : []
}
```

b) Zakupy

```
function addShopping(address, name){

    db.Shoppings.insert({
        ShopName: name,
        ShopAddress: address,
        Date: new Date(),
        Products: []
    })
}
addShopping("Kraków, Aleja Pokoju 11", "Media Markt");
addShopping("Kraków, Czarnowiejska 2", "Rossmann");
addShopping("Kraków, Kawiory 5", "Auchan");
```

```
/* 1 */
{
    "_id" : ObjectId("5e81e2ff7f4287f25112cd39"),
    "ShopName" : "Media Markt",
    "ShopAddress" : "Kraków, Aleja Pokoju 11",
    "Date" : ISODate("2020-03-30T12:15:59.146Z"),
    "Products" : []
}

/* 2 */
{
    "_id" : ObjectId("5e81e3637f4287f25112cd3a"),
    "ShopName" : "Rossmann",
    "ShopAddress" : "Kraków, Czarnowiejska 2",
    "Date" : ISODate("2020-03-30T12:17:39.085Z"),
    "Products" : []
}

/* 3 */
{
    "_id" : ObjectId("5e81e3637f4287f25112cd3b"),
    "ShopName" : "Auchan",
    "ShopAddress" : "Kraków, Kawiory 5",
    "Date" : ISODate("2020-03-30T12:17:39.086Z"),
    "Products" : []
}
```

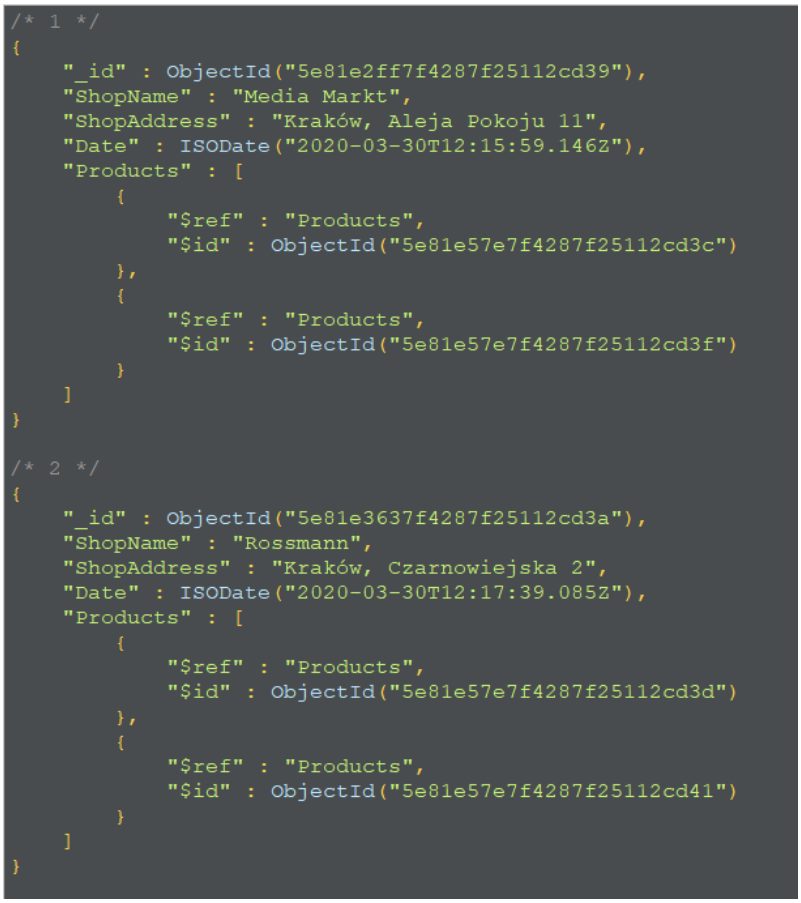
c) Produkty

```
function addProduct(name, category, price, description){

    db.Products.insert({
        Name: name,
        Category: category,
        Price: price,
        Description: description
    })
}
addProduct("Telewizor", "Elektronika", 2500, "Najnowszy model");
addProduct("Perfumy", "Uroda", 99.99, "Perfumy Calvin Klein");
addProduct("Chleb", "Żywność", 2.50, "Chleb żytni");
addProduct("Słuchawki", "Elektronika", 100.99, "Najlepszy dźwięk");
addProduct("Woda", "Żywność", 1.99, "Woda źródlana");
addProduct("Dezodorant", "Uroda", 12.99, "Odświeżający zapach");
```

```
/* 1 */
{
    "_id" : ObjectId("5e81e57e7f4287f25112cd3c"),
    "Name" : "Telewizor",
    "Category" : "Elektronika",
    "Price" : 2500.0,
    "Description" : "Najnowszy model"
}

/* 2 */
{
    "_id" : ObjectId("5e81e57e7f4287f25112cd3d"),
    "Name" : "Perfumy",
    "Category" : "Uroda",
    "Price" : 99.99,
    "Description" : "Perfumy Calvin Klein"
}

/* 3 */
{
    "_id" : ObjectId("5e81e57e7f4287f25112cd3e"),
    "Name" : "Chleb",
    "Category" : "Żywność",
    "Price" : 2.5,
    "Description" : "Chleb żytni"
}

/* 4 */
{
    "_id" : ObjectId("5e81e57e7f4287f25112cd3f"),
    "Name" : "Słuchawki",
    "Category" : "Elektronika",
    "Price" : 100.99,
    "Description" : "Najlepszy dźwięk"
}

/* 5 */
{
    "_id" : ObjectId("5e81e57e7f4287f25112cd40"),
    "Name" : "Woda",
    "Category" : "Żywność",
    "Price" : 1.99,
```

d) Dodanie przedmiotu do zakupów

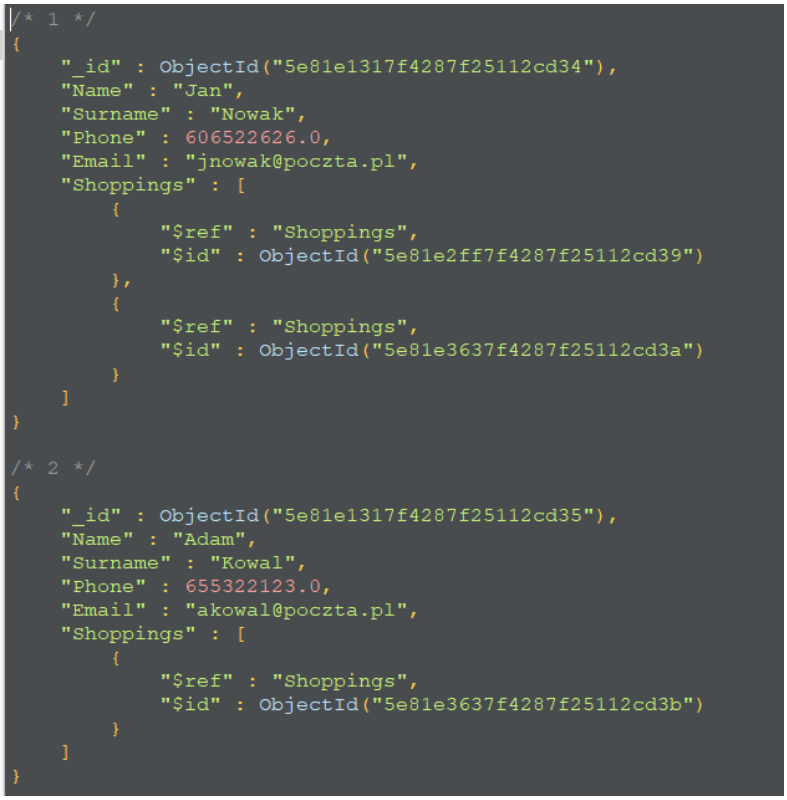
```
function addProductToShopping(shoppingID, productID){
    db.Shoppings.update(
        {_id: new ObjectId(shoppingID)},
        {$addToSet:{
            Products: {$ref : "Products", $id: new ObjectId(productID)}
        }
    });
}
addProductToShopping("5e81e2ff7f4287f25112cd39", "5e81e57e7f4287f25112cd3c")
addProductToShopping("5e81e2ff7f4287f25112cd39", "5e81e57e7f4287f25112cd3f")
addProductToShopping("5e81e3637f4287f25112cd3a", "5e81e57e7f4287f25112cd3d")
addProductToShopping("5e81e3637f4287f25112cd3a", "5e81e57e7f4287f25112cd41")
addProductToShopping("5e81e3637f4287f25112cd3b", "5e81e57e7f4287f25112cd3e")
addProductToShopping("5e81e3637f4287f25112cd3b", "5e81e57e7f4287f25112cd40")
```



```
/* 1 */
{
    "_id" : ObjectId("5e81e2ff7f4287f25112cd39"),
    "ShopName" : "Media Markt",
    "ShopAddress" : "Kraków, Aleja Pokoju 11",
    "Date" : ISODate("2020-03-30T12:15:59.146Z"),
    "Products" : [
        {
            "$ref" : "Products",
            "$id" : ObjectId("5e81e57e7f4287f25112cd3c")
        },
        {
            "$ref" : "Products",
            "$id" : ObjectId("5e81e57e7f4287f25112cd3f")
        }
    ]
}

/* 2 */
{
    "_id" : ObjectId("5e81e3637f4287f25112cd3a"),
    "ShopName" : "Rossmann",
    "ShopAddress" : "Kraków, Czarnowiejska 2",
    "Date" : ISODate("2020-03-30T12:17:39.085Z"),
    "Products" : [
        {
            "$ref" : "Products",
            "$id" : ObjectId("5e81e57e7f4287f25112cd3d")
        },
        {
            "$ref" : "Products",
            "$id" : ObjectId("5e81e57e7f4287f25112cd41")
        }
    ]
}
```

e) Dodanie zakupów do klienta

```
function addShoppingToClient(clientID, shoppingID){
    db.Clients.update(
        {_id: new ObjectId(clientID)},
        {$addToSet:
            {Shoppings:
                {$ref: "Shoppings",
                $id: new ObjectId(shoppingID)}
         }
     });
}

addShoppingToClient("5e81e1317f4287f25112cd34", "5e81e2ff7f4287f25112cd39")
addShoppingToClient("5e81e1317f4287f25112cd34", "5e81e3637f4287f25112cd3a")
addShoppingToClient("5e81e1317f4287f25112cd35", "5e81e3637f4287f25112cd3b")
```



```
/* 1 */
{
    "_id" : ObjectId("5e81e1317f4287f25112cd34"),
    "Name" : "Jan",
    "Surname" : "Nowak",
    "Phone" : 606522626.0,
    "Email" : "jnowak@poczta.pl",
    "Shoppings" : [
        {
            "$ref" : "Shoppings",
            "$id" : ObjectId("5e81e2ff7f4287f25112cd39")
        },
        {
            "$ref" : "Shoppings",
            "$id" : ObjectId("5e81e3637f4287f25112cd3a")
        }
    ]
}

/* 2 */
{
    "_id" : ObjectId("5e81e1317f4287f25112cd35"),
    "Name" : "Adam",
    "Surname" : "Kowal",
    "Phone" : 655322123.0,
    "Email" : "akowal@poczta.pl",
    "Shoppings" : [
        {
            "$ref" : "Shoppings",
            "$id" : ObjectId("5e81e3637f4287f25112cd3b")
        }
    ]
}
```