Bazy danych – NoSQL MongoDB – zadania

BARTŁOMIEJ PLEWNIA WTOREK 11.15A

Zad 1a.

Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (*business*). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

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

db.business.aggregate([
    {$group : {_id : "$city", val:{$sum: 1}}},
    {$sort : {_id: 1}}
    ])
```

0.022 sec.

```
/* 1 */
[
    "Ahwatukee",
    "Anthem",
    "Apache Junction",
    "Arcadia",
    "Atlanta",
    "Avondale",
    "Black Canyon City",
    "Bonnyrigg",
    "Boulder City",
    "Buckeye",
    "C Las Vegas",
    "Cambridge",
    "Carefree",
    "Casa Grande",
```

business 0.031 sec.

```
/* 1 */
{
    "_id" : "Ahwatukee",
    "val" : 8.0
}

/* 2 */
{
    "_id" : "Anthem",
    "val" : 63.0
}

/* 3 */
{
    "_id" : "Apache Junction",
    "val" : 91.0
```

Można zadanie wykonać dwoma poleceniami.

Zad 1b.

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

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

0.427 sec.

```
880318
```

Zad 1c.

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

```
db.business.aggregate([
    {$match: {open : false} },
    {$project : {name: true, full_address: true, stars: true} }
])
```

business 0.031 sec.


```
/* 1 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4eb"),
    "full_address" : "6401 University Ave\nMiddleton, WI 53562",
    "name" : "Crandalls Carryout & Catering",
    "stars" : 4.0
}

/* 2 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4f0"),
    "full_address" : "6230 University Ave\nMiddleton, WI 53562",
    "name" : "Mi Cocina",
    "stars" : 3.0
}

/* 3 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4f6"),
    "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
    "name" : "Charter Communications",
    "stars" : 1.5
}

/* 4 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf51a"),
    "full_address" : "6625 Century Ave\nMiddleton, WI 53562",
    "name" : "Stamm House At Pheasant Branch",
    "stars" : 2.0
}

/* 5 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf528"),
    "full_address" : "1901 Cayuga St\nMiddleton, WI 53562",
    "name" : "Soup Factory",
```

Zad 1d.

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

```
db.user.aggregate([
    {$match : {$or: [{'votes.funny' : 0}, {'votes.useful' : 0}] } },
    {$sort : {name: 1}}],
    {allowDiskUse: true}
    );
```

user 0.564 sec.


```
    "elite" : []
}

/* 39 */
{
    "_id" : ObjectId("5e79e3b48e663830c464e9dd"),
    "yelping_since" : "2013-09",
    "votes" : {
        "funny" : 0,
        "useful" : 0,
        "cool" : 0
    },
    "review_count" : 1,
    "name" : "A",
    "user_id" : "A3PDp2LWb5TNt-wO701e4w",
    "friends" : [],
    "fans" : 0,
    "average_stars" : 2.0,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}

/* 40 */
{
    "_id" : ObjectId("5e79e3b48e663830c464ea77"),
    "yelping_since" : "2012-09",
    "votes" : {
        "funny" : 0,
        "useful" : 2,
        "cool" : 0
    },
    "review_count" : 4,
    "name" : "A",
    "user_id" : "cvvA5vfM3Wv- ScjFTU36A",
```

Zad 1e.

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

```
db.tip.aggregate([
    {$match : {$and : [{date : {$gte : "2012-01-01"}}, {date : {$lt : "2013-01-01"}}] } },
    {$group : {_id : "$business_id", count: {$sum : 1}}},
    {$sort : {count: 1}}
]);
```

tip 0.275 sec.


```
/* 1 */
{
    "_id" : "CbUkl9BpdI_m7Yt4yTca_Q",
    "count" : 1.0
}

/* 2 */
{
    "_id" : "ToSm1CIH3xgaedt63f3FCQ",
    "count" : 1.0
}

/* 3 */
{
    "_id" : "uTvF-raBw4J9kp_C4jTSYA",
    "count" : 1.0
}

/* 4 */
{
    "_id" : "FHJQZ0p5ByTHpXUEOOB6Uw",
    "count" : 1.0
}

/* 5 */
{
    "_id" : "MHUX3HRj-gtHlg-YTbIVzQ",
    "count" : 1.0
}

/* 6 */
{
    "_id" : "fScnL-o3j2G9gUKHa3BaLg",
    "count" : 1.0
}
```

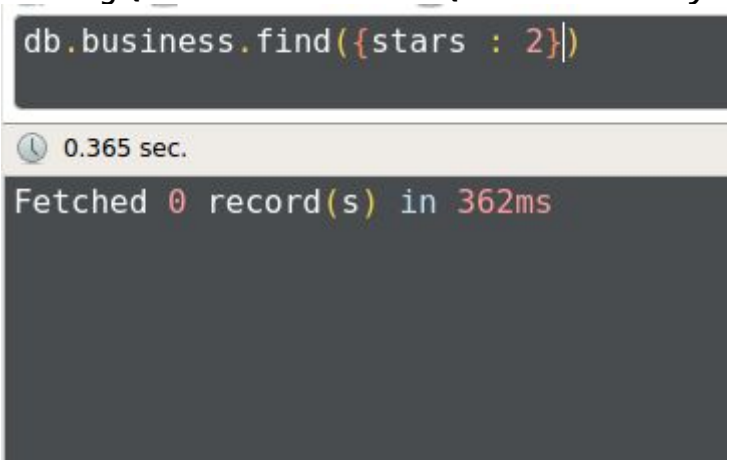
Zad 1f.

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

Zad 1g.

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

Próbując znaleźć usunięte dokumenty:



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

```
function insertReview(votesFunny, votesUseful, votesCool, user_id, review_id, stars, date, text, type, business_id){
    db.review.insert({
        votes : {
            funny : votesFunny,
            useful : votesUseful,
            cool : votesCool
        },
        user_id : user_id,
        review_id : review_id,
        stars : stars,
        date : {date:ISODate(date)},
        text : text,
        type : type,
        business_id : business_id
        })
}

insertReview(2,10,50,12,420,5,"2020-04-11","WHERE IS MY LAMB SAUCE?!","review","vcNAWiLM4dR7D2nwwJ7nCA")
```

```
0.055 sec.
Inserted 1 record(s) in 1ms
```

```
db.review.find({review_id : 420})
```

```
review  0.35 sec.
/* 1 */
{
    "_id" : ObjectId("5e9043559000eb3f2efb3400"),
    "votes" : {
        "funny" : 2.0,
        "useful" : 10.0,
        "cool" : 50.0
    },
    "user_id" : 12.0,
    "review_id" : 420.0,
    "stars" : 5.0,
    "date" : {
        "date" : ISODate("2020-04-11T00:00:00.000Z")
    },
    "text" : "WHERE IS MY LAMB SAUCE?!",
    "type" : "review",
    "business_id" : "vcNAWiLM4dR7D2nwwJ7nCA"
}
```

Zad 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.

```
function BusinessByCat(category){
    return db.business.find({categories : category},{categories : 1})
}


BusinessByCat("Restaurants")
```

business 0.003 sec.

```
/* 1 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4e3"),
    "categories" : [
        "American (Traditional)",
        "Restaurants"
    ]
}

/* 2 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4e4"),
    "categories" : [
        "Restaurants"
    ]
}

/* 3 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4e5"),
    "categories" : [
        "Chinese",
        "Restaurants"
    ]
}

/* 4 */
{
    "_id" : ObjectId("5e79e2f88e663830c44bf4e8"),
    "categories" : [
        "American (Traditional)",
        "Restaurants"
    ]
```

Dla pokazania wyników wyświetlam tylko atrybut categories

Zad 4.

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

```
db.user.find({user_id : "qtrmBGNqCvupHMHL_bKFgQ"})
```

user 0.088 sec.


```
/* 1 */
{
    "_id" : ObjectId("5e79e3b38e663830c4646ed3"),
    "yelping_since" : "2012-02",
    "votes" : {
        "funny" : 1,
        "useful" : 5,
        "cool" : 0
    },
    "review_count" : 6,
    "name" : "Lee",
    "user_id" : "qtrmBGNqCvupHMHL_bKFgQ",
    "friends" : [],
    "fans" : 0,
    "average_stars" : 3.83,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}
```


Przed update

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

updateUser("qtrmBGNqCvupHMHL_bKFgQ", "Robert")
```

0.001 sec.

```
Updated 1 existing record(s) in 1ms
```

Update

```
db.user.find({user_id : "qtrmBGNqCvupHMHL_bKFgQ"})
```

user 0.091 sec.


```
/* 1 */
{
    "_id" : ObjectId("5e79e3b38e663830c4646ed3"),
    "yelping_since" : "2012-02",
    "votes" : {
        "funny" : 1,
        "useful" : 5,
        "cool" : 0
    },
    "review_count" : 6,
    "name" : "Robert",
    "user_id" : "qtrmBGNqCvupHMHL_bKFgQ",
    "friends" : [],
    "fans" : 0,
    "average_stars" : 3.83,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}
```


Po update

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

```
var mapF = function(){
    emit(this.business_id, this.likes);
};

var reduceF = function(business_id, likes){
    return Array.avg(likes);
};

db.tip.mapReduce(
    mapF,
    reduceF,
    {out : "avg_tip_per_business"})

db.avg_tip_per_business.find()
```

avg_tip_per_business 0.001 sec.


```
    "value" : 0.0
}

/* 47 */
{
    "_id" : "-50sQyEuqRFKYTc1QHOZoQ",
    "value" : 0.0
}

/* 48 */
{
    "_id" : "-52_lPjA_YEInoyepTroWQ",
    "value" : 0.0
}

/* 49 */
{
    "_id" : "-52nir13iFnSMdBew674HA",
    "value" : 0.0
}

/* 50 */
{
    "_id" : "-584fn2GxYe9sLsgN2WeQA",
    "value" : 0.032258064516129
}
```

Zad 6a.

```java
public void task1a(){
    MongoCollection<Document> business = db.getCollection( s: "business");
    AggregateIterable<Document> cities = business.aggregate(Arrays.asList(
            Aggregates.group( id: "$city"),
            Aggregates.sort(Sorts.ascending( ...fieldNames: "_id"))));
    for(Document x: cities){
        System.out.println(x.getString( key: "_id")) ;
    }

}
```

MongoDB

```
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
Deforest
Eagan
Edinburgh
El Mirage
```

Zad 6b.

```
public void task2a(){
    MongoCollection<Document> review = db.getCollection( s: "review");
    System.out.println(review.countDocuments(Filters.gte( fieldName: "date",  value: "2011-01-01")));
}
```

```
880319

Process finished with exit code 0
```

Zad 6c.

```
public void task6c(){
    MongoCollection<Document> business = db.getCollection( s: "business");
    AggregateIterable<Document> closedBusinesses = business.aggregate(Arrays.asList(
            Aggregates.match(Filters.eq( fieldName: "open",  value: false)),
            Aggregates.project(Projections.include( ...fieldNames: "name", "full_address", "stars"))
    ));

    for(Document x: closedBusinesses){
        System.out.println(x.getString( key: "name") + ", " + x.getString( key: "full_address") + ", " +
                x.getDouble( key: "stars"));
    }
}
```

```
Ruen Thai, 7377 S Jones Blvd
Southwest
Las Vegas, NV 89139, 3.5
Tony's Bagel, Pizza & Deli, 2340 W Bell Rd
Phoenix, AZ 85023, 5.0
Dolce Salon & Spa At Lincoln, 6340 N Scottsdale Rd
Scottsdale, AZ 85253, 4.0
Stacy's Smoke Dem Bones Pit Stop BBQ, 1650 E Indian School Rd
Phoenix, AZ 85016, 3.0
Han's Chicken, 6850 Spring Mountain Rd
Ste F-6
Chinatown
Las Vegas, NV 89146, 3.5
```

Zad 6d.

```
public void task6d(){
    MongoCollection<Document> user = db.getCollection( s: "user");

    AggregateIterable<Document> users = user.aggregate(Arrays.asList(
            Aggregates.match(Filters.or(
                    Filters.eq( fieldName: "votes.funny",  value: 0),
                    Filters.eq( fieldName: "votes.useful",  value: 0)
            )),
            Aggregates.group( id: "$name"),
            Aggregates.sort(Sorts.ascending( ...fieldNames: "_id" ))));

    for(Document x: users){
        System.out.println(x) ;
    }
}
```

```
Document{{_id=A C}}
Document{{_id=A Castro}}
Document{{_id=A Famous}}
Document{{_id=A Great}}
Document{{_id=A Renee}}
Document{{_id=A Thomas}}
Document{{_id=A V}}
Document{{_id=A-ron}}
Document{{_id=A.}}
Document{{_id=A. D.}}
Document{{_id=A. J.}}
Document{{_id=A. Ralph}}
Document{{_id=A.J.}}
Document{{_id=A.M.}}
Document{{_id=A.P.}}
Document{{_id=A.joseph}}
Document{{_id=AA Printing}}
Document{{_id=AAA HOUSECLEANERS}}
Document{{_id=AARIN}}
Document{{_id=AB}}
Document{{_id=ABQ}}
Document{{_id=AC}}
Document{{_id=ACAP}}
Document{{_id=AD}}
Document{{_id=ADAM}}
Document{{_id=ADubz}}
Document{{_id=AE}}
Document{{_id=AGirlNamed}}
```

Zad 6e.

```
public void task6e(){
    MongoCollection<Document> tip = db.getCollection( s: "tip");

    AggregateIterable<Document> tips = tip.aggregate(Arrays.asList(
            Aggregates.match(Filters.and(
                    Filters.gte( fieldName: "date",  value: "2012-01-01"),
                    Filters.lt( fieldName: "date",  value: "2013-01-01")
            )),
            Aggregates.group( id: "$business_id", Accumulators.sum( fieldName: "nrOfTips",  expression: 1)),
            Aggregates.sort(Sorts.ascending( ...fieldNames: "nrOfTips" ))));

    for(Document x: tips){
        System.out.println(x) ;
    }
}
```

```
Document{{_id=FURgKkRFtMK5yKbjYZVVwA, nrOfTips=115}}
Document{{_id=lliksv-tglfUz1T3B3vgvA, nrOfTips=116}}
Document{{_id=xfwRO04KbAPw_zRotCfWQQ, nrOfTips=116}}
Document{{_id=kEC7OlpPnZRxCUyVwq7hig, nrOfTips=117}}
Document{{_id=6LM_Klmp3hOP0JmsMCKRqQ, nrOfTips=118}}
Document{{_id=4UVhuOLaMm2-34SrW8y-ag, nrOfTips=119}}
Document{{_id=_CHH9KN1bO5cs_GHjQ-r1A, nrOfTips=120}}
Document{{_id=sIyHTizqAiGu12XMLX3N3g, nrOfTips=121}}
Document{{_id=fwcoLKfZYMH6nZbAaeD6sw, nrOfTips=121}}
Document{{_id=YNQgak-ZLtYJQxlDwN-qIg, nrOfTips=122}}
Document{{_id=u9wjRhUjySkHPa_hG3kFOg, nrOfTips=123}}
Document{{_id=OdKCcB8skRPVSk7yBGwang, nrOfTips=124}}
Document{{_id=ME8gcI4BXjmFq7zEMtk3ag, nrOfTips=125}}
Document{{_id=l2lpBXx8jKl87J2szyJRuQ, nrOfTips=128}}
Document{{_id=GMDSE-m3yP0uCPfgkd8QSg, nrOfTips=131}}
Document{{_id=szw8OGJlsqaA3i2oe7dn9A, nrOfTips=137}}
Document{{_id=5paMI5Hhciyl09yGpzdoJw, nrOfTips=137}}
Document{{_id=JokKtdXU7zXHcr20Lrk29A, nrOfTips=139}}
Document{{_id=TWD8c5-P7w9v-2KX_GSNZQ, nrOfTips=147}}
Document{{_id=-7yF42k0CcJhtPw51oaOqQ, nrOfTips=148}}
Document{{_id=BqrTtox0JbG-P_DKBB5bBw, nrOfTips=150}}
Document{{_id=Es300Ys1XXPYg8aI7BKVYQ, nrOfTips=152}}
Document{{_id=Z0DD_sieK8ijAETu7OwGXw, nrOfTips=156}}
Document{{_id=34uJtlPnKicSaX1V8_tu1A, nrOfTips=158}}
Document{{_id=1fjgZaW6_qTYoAaSCfI-VQ, nrOfTips=158}}
Document{{_id=0UZ31UTcOLRKuqPqPe-VBA, nrOfTips=159}}
Document{{_id=uJYw4p59AKh8c8h5yWMdOw, nrOfTips=161}}
Document{{_id=PiV7phVMf7glDpH_J630Ow, nrOfTips=166}}
Document{{_id=pPfQ73--Y_h5Or0aqDgEGA, nrOfTips=166}}
Document{{_id=PXviRcHR1mqdH4vRc2LEAQ, nrOfTips=182}}
Document{{_id=z3SyT8blMIhsZNvKJgKcRA, nrOfTips=184}}
Document{{_id=eWPFXL1Bmu1ImtIa2Rqliw, nrOfTips=184}}
Document{{_id=H_SuH7uLiYahDMbNBB9kog, nrOfTips=185}}
```

Zad 6f.

```
public void task6f(){
    MongoCollection<Document> business = db.getCollection( s: "business");

    AggregateIterable<Document> avgStars = business.aggregate(Arrays.asList(
            Aggregates.match(Filters.gte( fieldName: "stars",  value: 4)),
            Aggregates.project(Projections.include( ...fieldNames: "name", "stars"))));

    for(Document x: avgStars){
        System.out.println(x) ;
    }
}
```

```
Document{{_id=5e79e2f98e663830c44c9951, name=Pro Nails By Andy, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9952, name=Sugar Free Salon, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9954, name=Vegas Bodyworks, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9956, name=Monsoon Siam, stars=4.5}}
Document{{_id=5e79e2f98e663830c44c9957, name=Jeremy House & The HOUSE Team, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c995a, name=Cream City Cafe, stars=4.5}}
Document{{_id=5e79e2f98e663830c44c995b, name=Tugas Amor, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c995d, name=Pastries N Chaat, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c995e, name=Glo Medspa, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c995f, name=Sorena Kebab House, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c9960, name=The Wiener's Circle Las Vegas, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c9961, name=The Charcoal Room, stars=4.5}}
Document{{_id=5e79e2f98e663830c44c9966, name=Giggles N Hugs, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9967, name=AllAces Plumbing, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9968, name=Chancery Lane Salon, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c996a, name=Stripchezze Food Truck, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c996d, name=Skinny Chick Fat Chicken, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c996e, name=Lish Nail Salon, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c996f, name=Lyft, stars=4.5}}
Document{{_id=5e79e2f98e663830c44c9970, name=Camelback CrossFit, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9971, name=Hooligans Bar & Grill, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9972, name=Garnachas Mexican Street Food, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c9973, name=Arrivederci Cucina Italiana, stars=4.0}}
Document{{_id=5e79e2f98e663830c44c9975, name=Carson Kitchen, stars=4.5}}
Document{{_id=5e79e2f98e663830c44c9976, name=Rustic Slice, stars=4.5}}
Document{{_id=5e79e2f98e663830c44c9977, name=Grinders Pizza Lounge, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c9979, name=Break Time Vape, stars=5.0}}
Document{{_id=5e79e2f98e663830c44c997a, name=Five Guys Burgers and Fries, stars=4.5}}
```

Zad 6g.

Aby sprawdzić poprawność wykonania zostaną usunięte wszystkie firmy z 3 gwiazdkami, ponieważ z 2 gwiazdkami zostały już wcześniej usunięte.

```java
public void task6g(){
    MongoCollection<Document> business = db.getCollection( s: "business");
    BasicDBObject query = new BasicDBObject("stars", 3);
    DeleteResult result = business.deleteMany(query);
    System.out.println("Number of deleted documents: " + result.getDeletedCount());
}
```

```
Number of deleted documents: 5754

Process finished with exit code 0
```

Zad 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

Tworzenie kolekcji

Dodanie przykładowych rzeczy

```
db.item.find()
```

item 0.008 sec.

```
/* 1 */
{
    "_id" : ObjectId("5e9196f99000eb3f2efb3401"),
    "name" : "carrot",
    "description" : "Good for your health",
    "price" : 420.0,
    "available" : true
}

/* 2 */
{
    "_id" : ObjectId("5e91970a9000eb3f2efb3402"),
    "name" : "cake",
    "description" : "Bad for your health",
    "price" : 123.0,
    "available" : false
}

/* 3 */
{
    "_id" : ObjectId("5e9197299000eb3f2efb3403"),
    "name" : "hammer",
    "description" : "Not for your health",
    "price" : 900.0,
    "available" : true
}
```

Dodanie przykładowych listy zakupów

```
db.shoppingList.find()
```

shoppingList 0.001 sec.

```
/* 1 */
{
    "_id" : ObjectId("5e9198549000eb3f2efb3404"),
    "itemsId" : [
        "5e9197299000eb3f2efb3403",
        "5e9196f99000eb3f2efb3401"
    ],
    "quantityOfItems" : [
        30.0,
        10.0
    ]
}

/* 2 */
{
    "_id" : ObjectId("5e9198a59000eb3f2efb3405"),
    "itemsId" : [
        "5e9196f99000eb3f2efb3401"
    ],
    "quantityOfItems" : [
        9.0
    ]
}
```

Dodanie przykładowych klientów

```
db.client.find()
```

client 0.002 sec.

```
/* 1 */
{
    "_id" : ObjectId("5e9199349000eb3f2efb3406"),
    "name" : "Krzysztof",
    "surname" : "Krawczyk",
    "address" : {
        "street" : "Poznanska",
        "house" : "13/2",
        "city" : "Los Angeles",
        "country" : "USA"
    },
    "shoppingListId" : [
        "5e9198549000eb3f2efb3404"
    ]
}

/* 2 */
{
    "_id" : ObjectId("5e9199659000eb3f2efb3407"),
    "name" : "Jędrzej",
    "surname" : "Wodzionka",
    "address" : {
        "street" : "Smoluchowskiego",
        "house" : "50",
        "city" : "Kraków",
        "country" : "Poland"
    },
    "shoppingListId" : []
}
```

Baza składa się z trzech kolekcji: client, shoppingList oraz item. Client posiada listę shoppingListId, w której przechowuje swoje zakupy. Natomiast shoppingList przechowuje dwie listy : listę produktów oraz odpowiadającą jej listę ilości zakupionych produktów.
Relacja przebiega następująco:
client ---> shoppingList ---> item