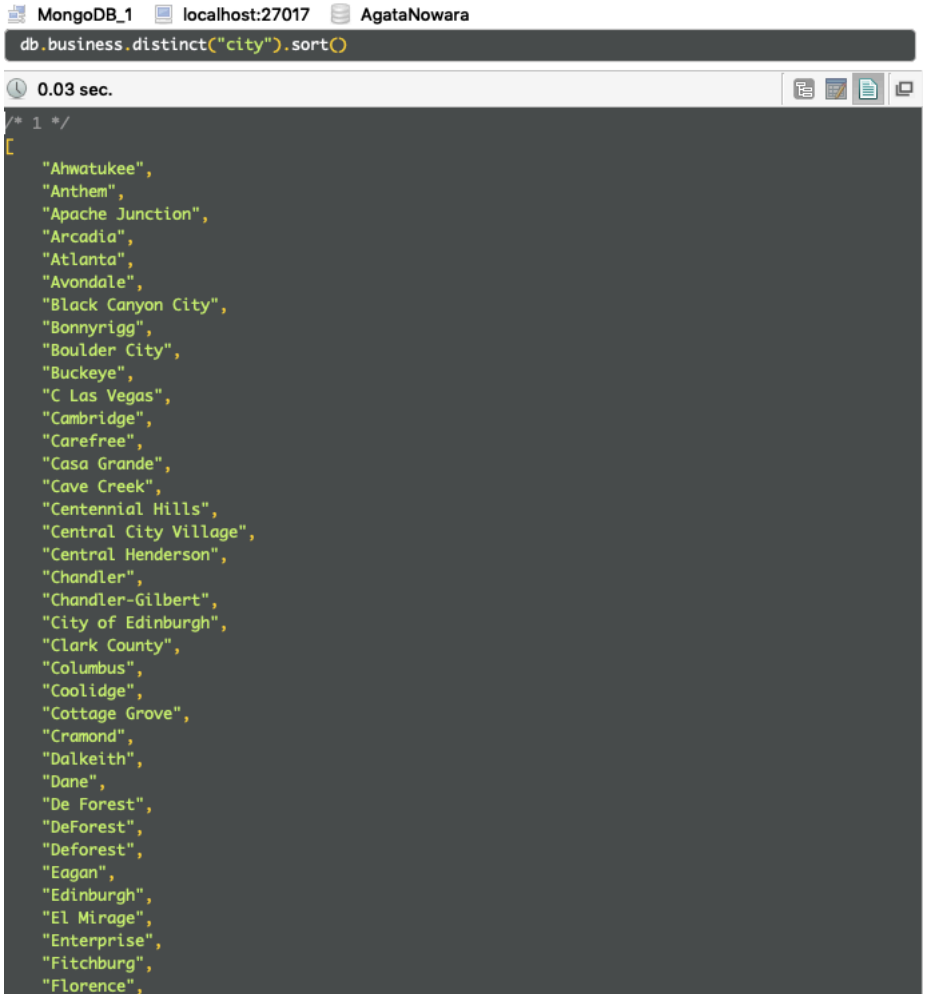
Zad 1

Polecenie: Wykorzystując bazę danych **yelp dataset** wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty:

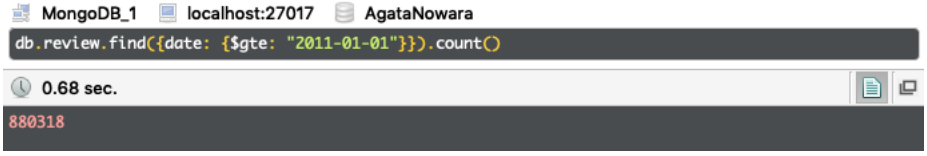
a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (*business*). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

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



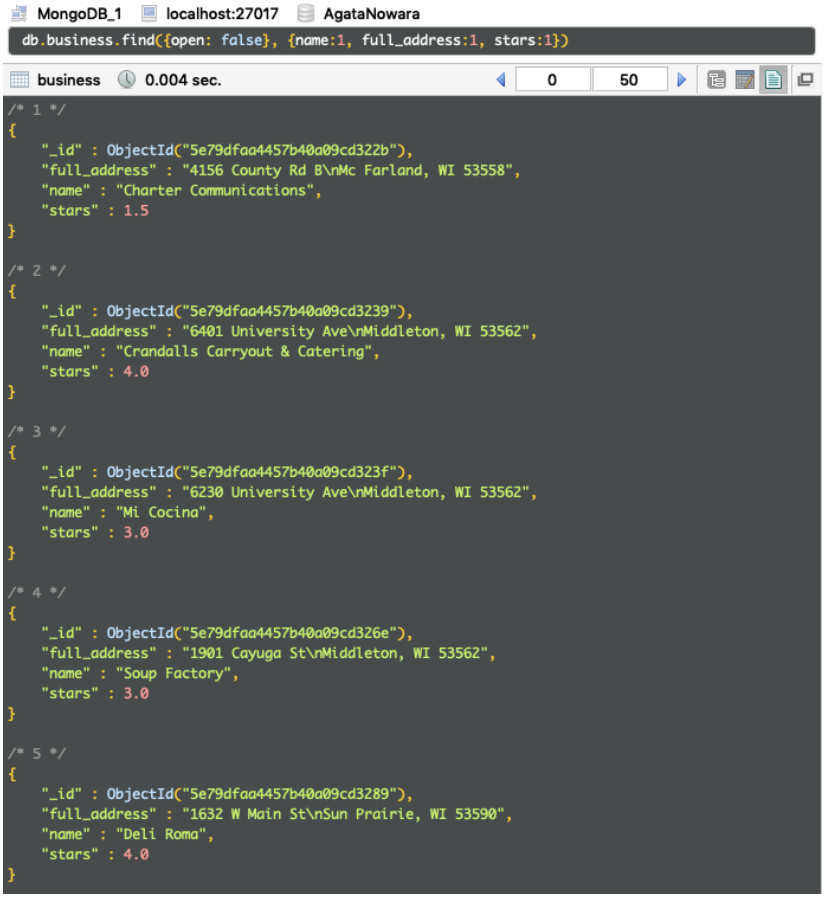
b. Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (włącznie).

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



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

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



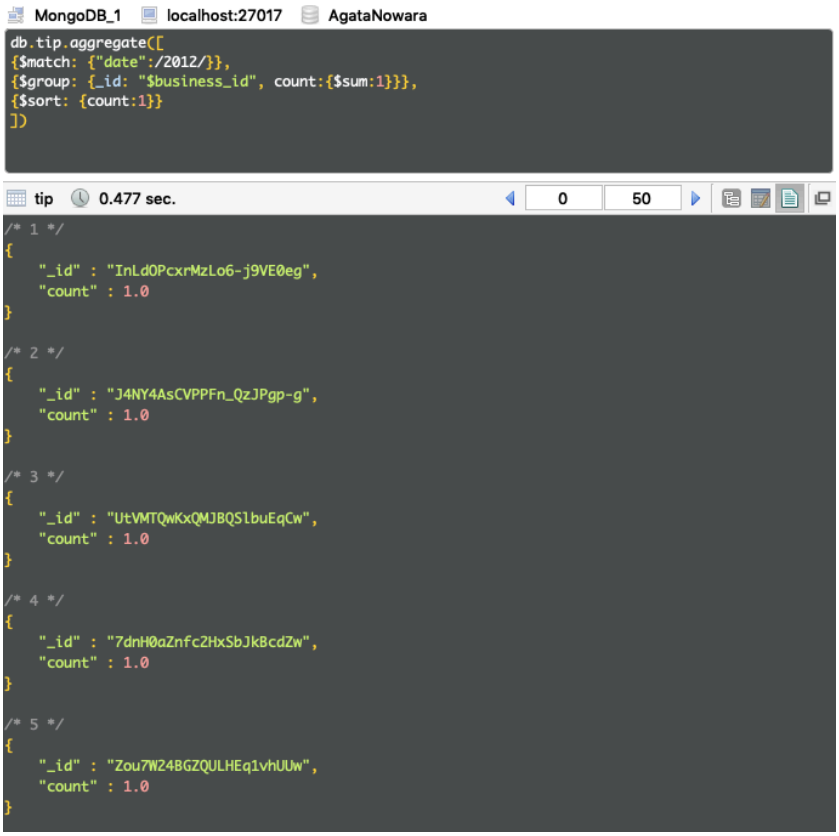
d. Zwróć dane wszystkich użytkowników (*user*), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (*funny lub useful*), wynik posortuj alfabetycznie według imienia użytkownika.

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

```
/* 2 */
{
    "_id" : ObjectId("5e79e17d870abad36ef05f41"),
    "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" : []
}

/* 3 */
{
    "_id" : ObjectId("5e79e17b870abad36eef2d38"),
    "yelping_since" : "2012-03",
    "votes" : {
        "funny" : 0,
        "useful" : 1,
        "cool" : 1
    },
    "review_count" : 2,
    "name" : "'Brandon",
    "user_id" : "CfWeBCqVvHze8bK_W5DcEg",
    "friends" : [
```

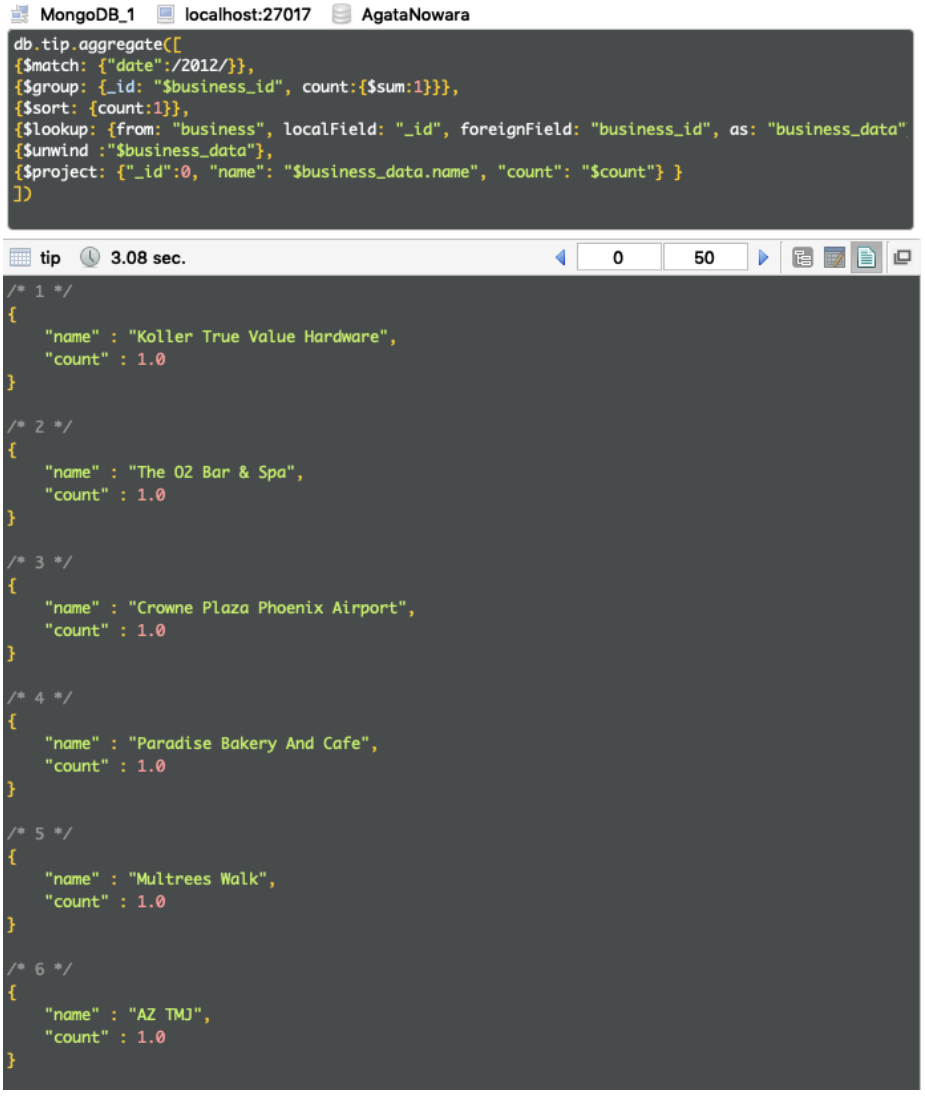

e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (*tip*) w 2012. Wynik posortuj alfabetycznie według liczby (*tip*).

```
db.tip.aggregate([
{$match: {"date":/2012/}},
{$group: {_id: "$business_id", count:{$sum:1}}},
{$sort: {count:1}}
])
```




```
/* 1 */
{
    "_id" : "InLdOPcxrMzLo6-j9VE0eg",
    "count" : 1.0
}

/* 2 */
{
    "_id" : "J4NY4AsCVPPFn_QzJPgp-g",
    "count" : 1.0
}

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

/* 4 */
{
    "_id" : "7dnH0aZnfc2HxSbJkBcdZw",
    "count" : 1.0
}

/* 5 */
{
    "_id" : "Zou7W24BGZQULHEq1vhUUw",
    "count" : 1.0
}
```

```
 db.tip.aggregate([
{$match: {"date":/2012/}},
{$group: {_id: "$business_id", count: {$sum:1}}},
{$sort: {count:1}},
{$lookup: {from: "business", localField: "_id", foreignField: "business_id",
 as: "business_data"}},
{$unwind: "$business_data"},
{$project: {"_id":0, "name": "$business_data.name", "count": "$count"}}
])
```



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([
{$match: {stars: {$gte: 4.0}}},
{$group: {_id: "$business_id", averageGrade: {$avg: "$stars"}}}
])
```

MongoDB_1 localhost:27017 AgataNowara

```
db.review.aggregate([
{$match: {stars: {$gte: 4.0}}},
{$group: {_id: "$business_id", averageGrade: {$avg: "$stars"}}}
])
```

review 1.66 sec. 0 50

```
/* 1 */
{
    "_id" : "aAfkYj5DacC91NqPpU4iow",
    "averageGrade" : 4.0
}

/* 2 */
{
    "_id" : "RtsSmkgNtUhWrJq6YsEfPw",
    "averageGrade" : 4.0
}

/* 3 */
{
    "_id" : "cc9KFNrcY9gA7t9D1a3FpA",
    "averageGrade" : 4.52727272727273
}

/* 4 */
{
    "_id" : "wn7GiuOQ_0VrPXNc1wVTDA",
    "averageGrade" : 4.33333333333333
}

/* 5 */
{
    "_id" : "N1PH93YA6gzJVXuYQ3LLRw",
    "averageGrade" : 4.6
}
```

```
db.review.aggregate([
{$match: {stars: {$gte: 4.0}}},
{$group: {_id: "$business_id", averageGrade: {$avg: "$stars"}}},
{$lookup: {from: "business", localField: "_id", foreignField: "business_id",
as: "business_data"}},
{$project: {"_id":0, "name": "$business_data.name", "averageGrade":
"$averageGrade"}}
])
```

MongoDB_1 localhost:27017 AgataNowara

```
db.review.aggregate([
{$match: {stars: {$gte: 4.0}}},
{$group: {_id: "$business_id", averageGrade: {$avg: "$stars"}}},
{$lookup: {from: "business", localField: "_id", foreignField: "business_id", as: "business_data"}},
{$unwind: "$business_data"},
{$project: {"_id":0, "name": "$business_data.name", "averageGrade": "$averageGrade"}}
])
```

review 4.36 sec. 0 50

```
/* 1 */
{
    "name" : "Sugar Factory",
    "averageGrade" : 4.3719298245614
}

/* 2 */
{
    "name" : "Vans",
    "averageGrade" : 5.0
}

/* 3 */
{
    "name" : "International Auto Painting & Body",
    "averageGrade" : 5.0
}

/* 4 */
{
    "name" : "Rock the Tea",
    "averageGrade" : 4.66666666666667
}

/* 5 */
{
    "name" : "Harlan Sparer",
    "averageGrade" : 5.0
}
```

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

```
db.business.find({stars:2.0}).count()
```


```
db.business.remove({stars:2.0})
```

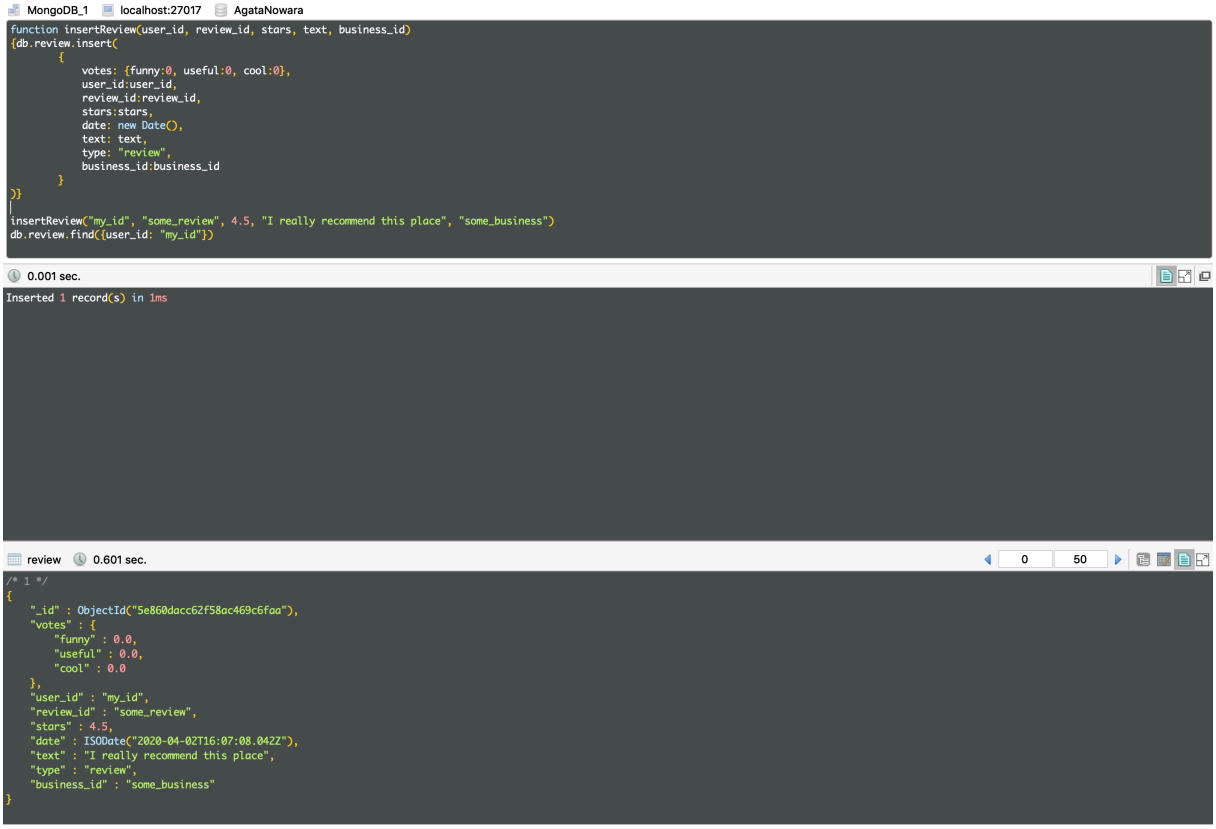

```
db.business.find({stars:2.0}).count()
```

Zad 2

Polecenie: Zdefiniuj funkcję (*MongoDB*) umożliwiającą dodanie nowej recenzji (*review*). Wykonaj przykładowe wywołanie.

```
 function insertReview(user_id, review_id, stars, text, business_id)

function insertReview(user_id, review_id, stars, text, business_id)

{db.review.insert(

        {

            votes: {funny:0, useful:0, cool:0},

            user_id:user_id,

            review_id:review_id,

            stars:stars,

            date: new Date(),

            text: text,

            type: "review",

            business_id:business_id

        }

)}

insertReview("my_id", "some_review", 4.5, "I really recommend this place", "some_business")
```

```
db.review.find({user_id: "my_id"})
```



Zad 3

Polecenie: 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 findBusinessesFromCategory(category)

{

    return db.business.find({categories: {$regex: category}})

}

findBusinessesFromCategory("Event")
```

Zad 4

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

```
function renameUser(user_id, new_name)

{

    db.user.update(

        {user_id: user_id},

        {$set: {name:new_name}},

        {multi:true}

     )

}

renameUser("MWhR9LvOdRbqtu1I_DRFBg", "Monitor")

db.user.find({name:"Monitor"})
```

MongoDB_1 localhost:27017 AgataNowara

```
function renameUser(user_id, new_name)
{
    db.user.update(
        {user_id: user_id},
        {$set: {name:new_name}},
        {multi:true}
    )
}
renameUser("MWhR9LvOdRbqtu1I_DRFBg", "Monitor")
db.user.find({name:"Monitor"})
```

0.149 sec.

```
Updated 1 existing record(s) in 149ms
```

user 0.133 sec. 0 50


```
/* 1 */
{
    "_id" : ObjectId("5e79e17a870abad36eeea107"),
    "yelping_since" : "2011-12",
    "votes" : {
        "funny" : 0,
        "useful" : 0,
        "cool" : 0
    },
    "review_count" : 1,
    "name" : "Monitor",
    "user_id" : "MWhR9LvOdRbqtu1I_DRFBg",
    "friends" : [
        "8Y2EN4XNNhnwssuPb31sJg",
        "A1jPleJ99kXZ3t9wQ3np-g",
        "resYiOoGkQg6q0qgtj_1GA",
        "skl1OnkjMqD4GdFpVhU88Q",
        "3Ss2aqrSoO7WbxE2GcLlfQ",
        "SGy3JDbhtzDTTBO7unqQxg",
        "Aez4Y1F0m2ucIfmfzPZfjw",
        "fT4jx_9cRWyKFROxv_MIrw",
        "QmZOAYM7ITbTdkTn6ay_ag",
        "_l18_te9ifeArhNdCfFy1g",
        "H6ovBilqI3cSunx27XbPUg",
        "A8pmiEiiWjXci_oeigLNZQ",
        "fdSfg45S-P-FcVF-VIUOew"
    ],
    "fans" : 0,
    "average_stars" : 5.0,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}
```



```
{
    "_id" : ObjectId("5e79e17a870abad36eeea107"),
    "yelping_since" : "2011-12",
    "votes" : {
        "funny" : 0,
        "useful" : 0,
        "cool" : 0
    },
    "review_count" : 1,
    "name" : "Matthew",
    "user_id" : "MWhR9LvOdRbqtu1I_DRFBg",
    "friends" : [
        "8Y2EN4XNNhnwssuPb31sJg",
        "A1jPleJ99kXZ3t9wQ3np-g",
        "resYiOoGkQg6q0qgtj_1GA",
        "skl1OnkjMqD4GdFpVhU88Q",
        "3Ss2aqrSoO7WbxE2GcLlfQ",
        "SGy3JDbhtzDTTBO7unqQxg",
        "Aez4Y1F0m2ucIfmfzPZfjw",
        "fT4jx_9cRWyKFROxv_MIrw",
        "QmZOAYM7ITbTdkTn6ay_ag",
        "_l18_te9ifeArhNdCfFy1g",
        "H6ovBilqI3cSunx27XbPUg",
        "A8pmiEiiWjXci_oeigLNZQ",
        "fdSfg45S-P-FcVF-VIUOew"
    ],
    "fans" : 0,
    "average_stars" : 5.0,
    "type" : "user",
    "compliments" : {},
    "elite" : []
}
```


Zad 5

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

```
var mapFunction = function()
{
    var key = this.business_id;
    var value = { sum: this.likes, count: 1 };
    emit(key, value);
};
var reduceFunction = function(key, values)
{
    var result = { sum: 0, count: 0 };
    values.forEach(
        function(value)
        {
            result.sum += value.sum;
            result.count += value.count;
         }
    )
    return result;
}
var finalizeFunction = function(key, reducedValue)
{
    var avgTips = reducedValue.sum / reducedValue.count;
    return avgTips;
}
db.tip.mapReduce(
    mapFunction,
    reduceFunction,
    {
        out: "tips_per_business_avg",
        finalize: finalizeFunction
    })
```

```
db.tips_per_business_avg.find()
```

Zad 6

Polecenie: Odwzoruj wszystkie zadania z punktu 1 w języku programowania (np. JAVA) z pomocą API do MongoDB. Wykorzystaj dla każdego zadania odrębną metodę.

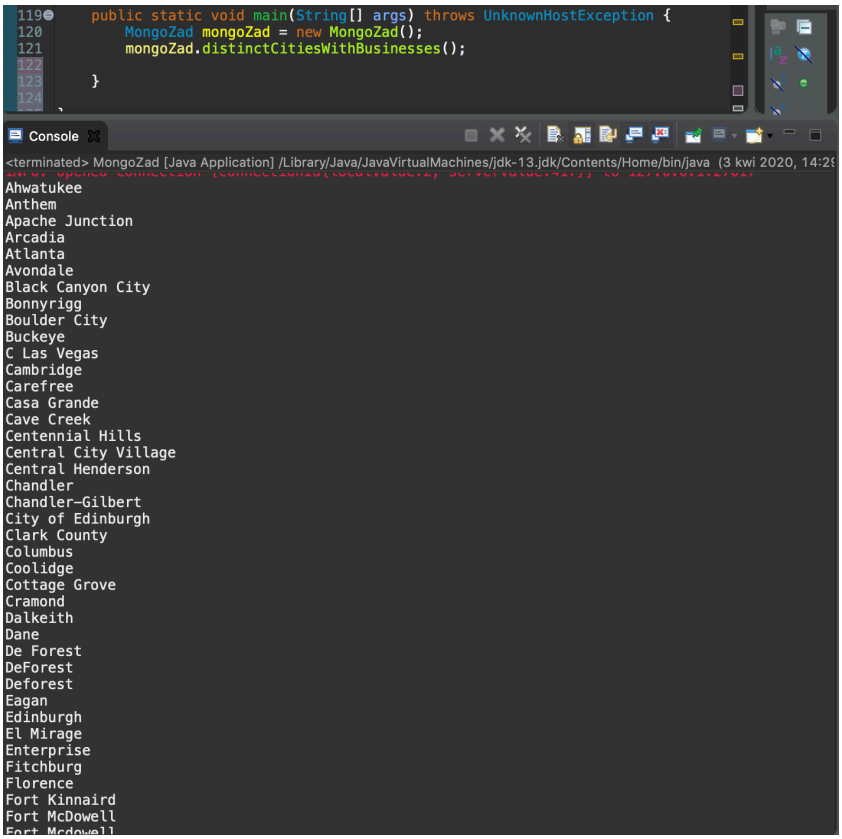
```
package mongo_zad;

import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.Arrays;

import com.mongodb.*;
import com.mongodb.client.AggregateIterable;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Accumulators;
import com.mongodb.client.model.Aggregates;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Projections;
import com.mongodb.client.model.Sorts;
import com.mongodb.client.MongoCursor;
import org.bson.Document;
import org.bson.conversions.Bson;

public class MongoZad {
	private MongoClient mongoClient;
	private MongoDatabase db;

	public MongoZad() throws UnknownHostException {
		mongoClient = new MongoClient();
		db = mongoClient.getDatabase("AgataNowara");
	}


	//1a

	public void distinctCitiesWithBusinesses()
	{
		MongoCollection<Document>business=db.getCollection("business");
		MongoCursor<String> cities = business.distinct("city", String.class).iterator();
		while(cities.hasNext())
		{
			System.out.println(cities.next());
		}
	}

	//1b

	public long countReviewsAfter2011()
	{
		MongoCollection<Document>reviews=db.getCollection("review");
		long counter=reviews.countDocuments(Filters.gte("date","2011-01-01"));
		System.out.println(counter);
		return counter;
	}

	//1c
	public void findClosedBusinesses()
	{
		MongoCollection<Document> business = db.getCollection("business");
		Bson bsonFilter = Filters.eq("open", false);
		FindIterable<Document> closed_businesses =
business.find(bsonFilter).projection(Projections.include("name","full_address", "stars" ));
		MongoCursor<Document> businesses_list=closed_businesses.iterator();
		while(businesses_list.hasNext())
```

```
                {
                        System.out.println(businesses_list.next());
                }

        }

        //1d
        public void findNotFunnyOrNotUsefull()
        {
                MongoCollection<Document>users = db.getCollection("user");
                FindIterable<Document> chosen_users = users.find(Filters.or(Filters.eq("votes.funny",
0), Filters.eq("votes.usefull", 0))).sort(Sorts.ascending("name"));
                MongoCursor<Document> iterableUsers=chosen_users.iterator();
                while(iterableUsers.hasNext())
                {
                        System.out.println(iterableUsers.next());
                }
        }

        //1e
        public void countTipsPerBusiness()
        {
                MongoCollection<Document> tips = db.getCollection("tip");
                AggregateIterable aggregationResult = tips.aggregate(Arrays.asList(
                                Aggregates.match(Filters.regex("date", "2012")),
                                Aggregates.group("$business_id", Accumulators.sum("count", 1)),
                                Aggregates.sort(Sorts.ascending("count")),
                                Aggregates.lookup("business", "_id", "business_id", "business_data"),
                                Aggregates.unwind("$business_data"),
                                Aggregates.project(Projections.include("business_data.name",
"count")),
                                Aggregates.project(Projections.exclude("_id"))
                ));
                for(Object result: aggregationResult)
                {
                        System.out.println(result);
                }
        }

        //1f

        public void countAvgBusinessGrade()
        {
                MongoCollection<Document>review = db.getCollection("review");
                AggregateIterable aggregationResult = review.aggregate(Arrays.asList(
                                Aggregates.match(Filters.gte("stars",4.0)),
                                Aggregates.group("$business_id", Accumulators.avg("avgStars",
"$stars")),
                                Aggregates.lookup("business", "_id", "business_id", "business_data"),
                                Aggregates.unwind("$business_data"),
                                Aggregates.project(Projections.include("business_data.name",
"avgStars")),
                                Aggregates.project(Projections.exclude("_id"))
                                ));
                for(Object result: aggregationResult)
                {
                        System.out.println(result);
                }
        }

        //1g

        public void remove2StarsBusinesses()
        {
                db.getCollection("businesses").deleteMany(Filters.eq("stars", 2));
        }

        public long count2StarsBusinesses()
        {
                MongoCollection<Document> business = db.getCollection("business");
                long counter = business.countDocuments(Filters.eq("stars", 2));
                return counter;
        }
```

```
        public static void main(String[] args) throws UnknownHostException {
                MongoZad mongoZad = new MongoZad();
//              mongoZad.distinctCitiesWithBusinesses();
//              mongoZad.countReviewsAfter2011();
//              mongoZad.findClosedBusinesses();
//              mongoZad.findNotFunnyOrNotUsefull();
//              mongoZad.countTipsPerBusiness();
                mongoZad.remove2StarsBusinesses();
                System.out.println(mongoZad.count2StarsBusinesses());

        }

}
```

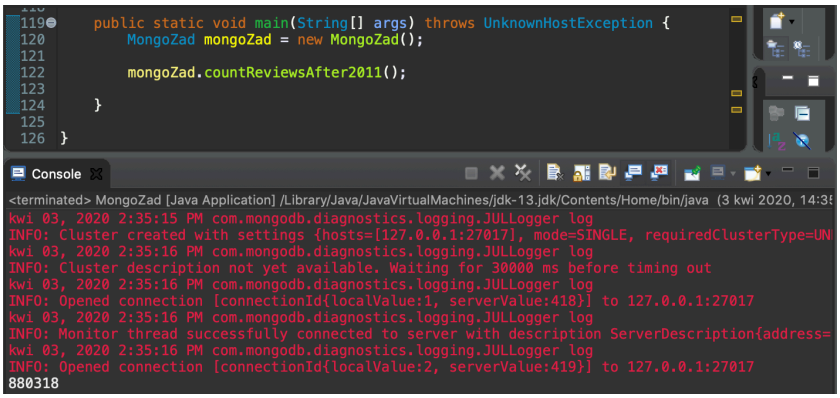
a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (*business*). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
public void distinctCitiesWithBusinesses()
        {
                MongoCollection<Document>business=db.getCollection("business");
                MongoCursor<String> cities = business.distinct("city",
String.class).iterator();
                while(cities.hasNext())
                {
                        System.out.println(cities.next());
                }

        }
```



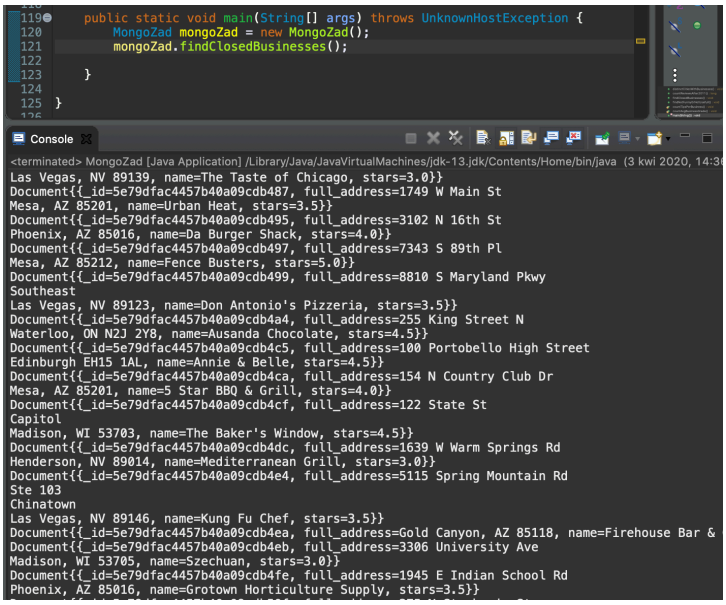
b. Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (włącznie).

```
public long countReviewsAfter2011()
	{
		MongoCollection<Document>reviews=db.getCollection("review");
		long counter=reviews.countDocuments(Filters.gte("date","2011-01-01"));
		System.out.println(counter);
		return counter;
	}
```



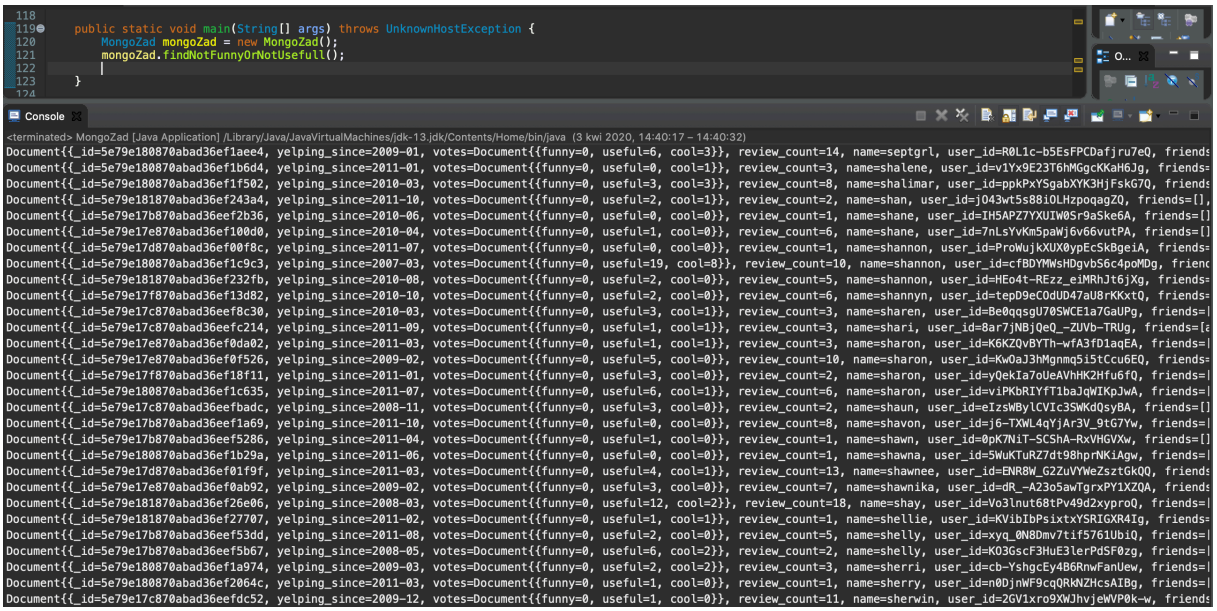
c. Zwróć dane wszystkich zamkniętych (*open*) firm (*business*) z pól: nazwa, adres, gwiazdki (*stars*).

```
public void findClosedBusinesses()
	{
		MongoCollection<Document> business = db.getCollection("business");
		Bson bsonFilter = Filters.eq("open", false);
		FindIterable<Document> closed_businesses =
business.find(bsonFilter).projection(Projections.include("name","full_address", "stars" ));
		MongoCursor<Document> businesses_list=closed_businesses.iterator();
		while(businesses_list.hasNext())
		{
			System.out.println(businesses_list.next());
		}

	}
```



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

```
public void findNotFunnyOrNotUsefull()
{
	MongoCollection<Document>users = db.getCollection("user");
	FindIterable<Document> chosen_users =
users.find(Filters.or(Filters.eq("votes.funny", 0), Filters.eq("votes.usefull",
0))).sort(Sorts.ascending("name"));
	MongoCursor<Document> iterableUsers=chosen_users.iterator();
	while(iterableUsers.hasNext())
	{
		System.out.println(iterableUsers.next());
	}
}
```



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

```
public void countTipsPerBusiness()
	{
		MongoCollection<Document> tips = db.getCollection("tip");
		AggregateIterable aggregationResult = tips.aggregate(Arrays.asList(
					Aggregates.match(Filters.regex("date", "2012")),
					Aggregates.group("$business_id", Accumulators.sum("count",
1)),
					Aggregates.sort(Sorts.ascending("count")),
					Aggregates.lookup("business", "_id", "business_id",
"business_data"),
					Aggregates.unwind("$business_data"),
					Aggregates.project(Projections.include("business_data.name",
"count")),
					Aggregates.project(Projections.exclude("_id"))
		));
		for(Object result: aggregationResult)
		{
			System.out.println(result);
		}
	}
```

```
118
119●    public static void main(String[] args) throws UnknownHostException {
120         MongoZad mongoZad = new MongoZad();
121         mongoZad.countTipsPerBusiness();
122
123     }
```

Console

```
MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java  (3 kwi 2020, 14:42:55)
Document{{count=1, business_data=Document{{name=Koller True Value Hardware}}}}
Document{{count=1, business_data=Document{{name=The O2 Bar & Spa}}}}
Document{{count=1, business_data=Document{{name=Crowne Plaza Phoenix Airport}}}}
Document{{count=1, business_data=Document{{name=Paradise Bakery And Cafe}}}}
Document{{count=1, business_data=Document{{name=Multrees Walk}}}}
Document{{count=1, business_data=Document{{name=AZ TMJ}}}}
Document{{count=1, business_data=Document{{name=Asi Es My Tierra}}}}
Document{{count=1, business_data=Document{{name=B & L Pool Supply}}}}
Document{{count=1, business_data=Document{{name=Ocotillo Dental Care}}}}
Document{{count=1, business_data=Document{{name=Timbers - Lake Mead}}}}
Document{{count=1, business_data=Document{{name=Papa John's Pizza}}}}
Document{{count=1, business_data=Document{{name=Simon Kitchen and Bar}}}}
Document{{count=1, business_data=Document{{name=RUB Sports Grill}}}}
Document{{count=1, business_data=Document{{name=Arizona Country Club}}}}
Document{{count=1, business_data=Document{{name=Hilberto's Mexican Food}}}}
Document{{count=1, business_data=Document{{name=Barrett-Jackson Auction Company LLC}}}}
Document{{count=1, business_data=Document{{name=Fantastic Sam's}}}}
Document{{count=1, business_data=Document{{name=McDonald's}}}}
Document{{count=1, business_data=Document{{name=Szechuan}}}}
Document{{count=1, business_data=Document{{name=Enterprise Rent-A-Car}}}}
Document{{count=1, business_data=Document{{name=Arizona Department of Environmental Quality}}}}
Document{{count=1, business_data=Document{{name=O'reilly Auto Parts}}}}
Document{{count=1, business_data=Document{{name=Scott's Deli}}}}
Document{{count=1, business_data=Document{{name=La Cerise}}}}
Document{{count=1, business_data=Document{{name=Water and Ice Discount Superstores}}}}
Document{{count=1, business_data=Document{{name=Port of Subs}}}}
Document{{count=1, business_data=Document{{name=Chase}}}}
Document{{count=1, business_data=Document{{name=Korea House BBQ}}}}
Document{{count=1, business_data=Document{{name=Baskin-Robbins}}}}
Document{{count=1, business_data=Document{{name=The Home Depot}}}}
Document{{count=1, business_data=Document{{name=Quality Inn & Suites of the Sun Cities}}}}
```

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

```
public void countAvgBusinessGrade()
{
        MongoCollection<Document>review = db.getCollection("review");
        AggregateIterable aggregationResult = review.aggregate(Arrays.asList(
                        Aggregates.match(Filters.gte("stars",4.0)),
                        Aggregates.group("$business_id", Accumulators.avg("avgStars",
"$stars")),
                        Aggregates.lookup("business", "_id", "business_id",
"business_data"),
                        Aggregates.unwind("$business_data"),
                        Aggregates.project(Projections.include("business_data.name",
"avgStars")),
                        Aggregates.project(Projections.exclude("_id"))
                        ));
        for(Object result: aggregationResult)
        {
                System.out.println(result);
        }
}
```

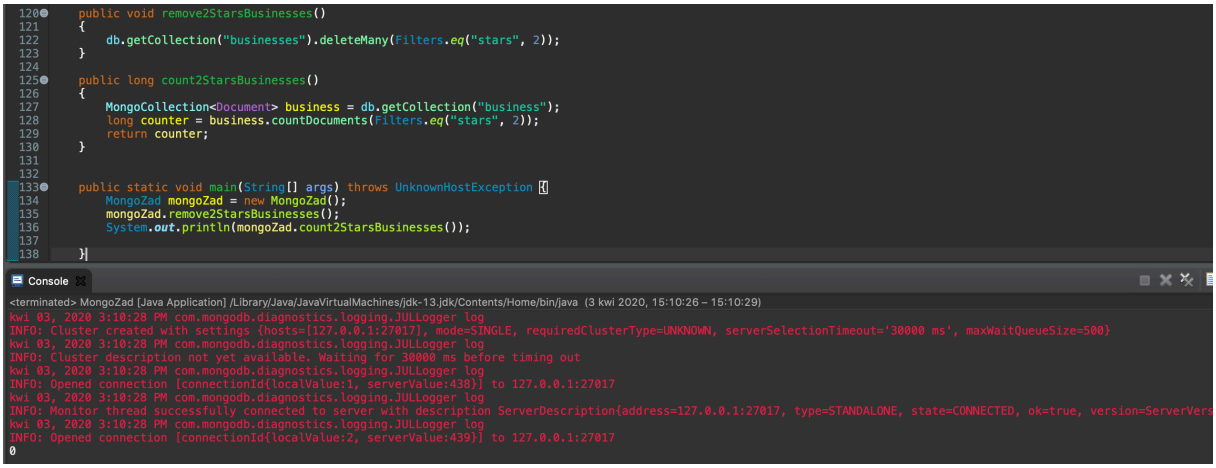
```
118
119●    public static void main(String[] args) throws UnknownHostException {
120         MongoZad mongoZad = new MongoZad();
121         mongoZad.countAvgBusinessGrade();
122     }
```

Console

```
MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java  (3 kwi 2020, 14:54:22)
Document{{avgStars=5.0, business_data=Document{{name=Vans}}}}
Document{{avgStars=5.0, business_data=Document{{name=International Auto Painting & Body}}}}
Document{{avgStars=4.666666666666667, business_data=Document{{name=Rock the Tea}}}}
Document{{avgStars=5.0, business_data=Document{{name=Harlan Sparer}}}}
Document{{avgStars=4.0, business_data=Document{{name=WH Smith}}}}
Document{{avgStars=5.0, business_data=Document{{name=ROC2 Fresh Roasted Organic Coffee}}}}
Document{{avgStars=4.375, business_data=Document{{name=Arizona Taco  Festival}}}}
Document{{avgStars=4.4324324324324325, business_data=Document{{name=El Norteño}}}}
Document{{avgStars=5.0, business_data=Document{{name=One Wing Boutique}}}}
Document{{avgStars=4.2, business_data=Document{{name=Port of Subs}}}}
Document{{avgStars=5.0, business_data=Document{{name=Maui Saito Ninjitsu}}}}
Document{{avgStars=4.333333333333333, business_data=Document{{name=Lyte Lounge & Bistro}}}}
Document{{avgStars=4.0, business_data=Document{{name=Albertsons}}}}
Document{{avgStars=4.5, business_data=Document{{name=The Spa}}}}
Document{{avgStars=4.0, business_data=Document{{name=Student Recreation Complex}}}}
Document{{avgStars=4.5, business_data=Document{{name=7-Eleven}}}}
Document{{avgStars=4.857142857142857, business_data=Document{{name=Pampered and Polished Nails}}}}
Document{{avgStars=4.666666666666667, business_data=Document{{name=Huger Mercy Living Center}}}}
Document{{avgStars=4.384615384615385, business_data=Document{{name=Peoria Cafe}}}}
Document{{avgStars=4.333333333333333, business_data=Document{{name=Firehouse Subs}}}}
Document{{avgStars=4.4, business_data=Document{{name=Scottsdale Salt River}}}}
Document{{avgStars=4.0, business_data=Document{{name=Miam Miam}}}}
Document{{avgStars=4.666666666666667, business_data=Document{{name=Taste & Sounds of Soul}}}}
Document{{avgStars=4.25, business_data=Document{{name=Billet Bar}}}}
Document{{avgStars=4.666666666666667, business_data=Document{{name=Subway}}}}
Document{{avgStars=4.5, business_data=Document{{name=Dunkin' Donuts}}}}
```

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

```
public void remove2StarsBusinesses()
    {
            db.getCollection("businesses").deleteMany(Filters.eq("stars", 2));
    }

    public long count2StarsBusinesses()
    {
            MongoCollection<Document> business = db.getCollection("business");
            long counter = business.countDocuments(Filters.eq("stars", 2));
            return counter;
    }
```



Zad 7

Polecenie: 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

Przygotowanie dokumentów json dla 3 kolekcji wraz z przykładowymi danymi

Users > agata > {} customer.json > ...

```
1  {"customer_id": "customerID1", "name": "Agata", "surname": "Bubik", "full_address": {"city":"Katowice","street": "Miła" ,
   "number": 21, "postal_code": "40-371"}, "e-mail": "ab@onet.pl", "phone": "603405671"}
2  {"customer_id": "customerID2", "name": "Maciej", "surname": "Mitser", "full_address": {"city":"Warszawa", "street":
   "Sezamkowa" ,"number": 4, "postal_code": "30-562"}, "e-mail": "mm@gmail.pl", "phone": "708623608"}
3  {"customer_id": "customerID3", "name": "Julia", "surname": "Suchy", "full_address": {"city":"Rzeszów","street": "Nad
   Jarem" ,"number": 7, "postal_code": "30-308"}, "e-mail": "js@onet.pl", "phone": "603405671"}
```

Users > agata > {} purchase.json > ...

```
1  {"order_id": "orderID1", "customer_id" : "customerID1", "date": "2020-04-01", "products": [{"product_id": "productID1",
   "quantity": 1, "discount": 0.1}]} {"order_id": "orderID2", "customer_id" : "customerID2", "date": "2020-04-02",
   "products": [{"product_id": "productID1", "quantity": 1, "discount": 0.1}, {"product_id": "productID3", "quantity": 2,
   "discount": 0.3}]} {"order_id": "orderID3", "customer_id" : "customerID3", "date": "2020-04-03", "products": [
   {"product_id": "productID2", "quantity": 2, "discount": 0.2},{"product_id": "productID3", "quantity": 1, "discount": 0.1} ]
   }
```

Users > agata > {} product.json > ...

```
1  {"product_id": "productID1", "category": "bag", "brand": "Gucci",  "name": "bag1", "price": 1279, "quantity_in_stock": 20}
2  {"product_id": "productID2", "category": "sunglasses", "brand": "Rayban", "name": "ClubMaster", "price": 699,
   "quantity_in_stock": 40}
3  {"product_id": "productID3", "category": "sport equipment","brand": "Gregory", "name": "backpack1", "price": 699,
   "quantity_in_stock": 30}
```

Zaimportowanie wcześniej utworzonych dokumentów do bazy Shop:

```
MacBook-Pro-Agata:~ agata$ mongoimport --db Shop --collection customer --type json --file /Users/agata/customer.json
2020-04-04T14:00:55.531+0200    connected to: mongodb://localhost/
2020-04-04T14:00:55.571+0200    3 document(s) imported successfully. 0 document(s) failed to import.
MacBook-Pro-Agata:~ agata$ mongoimport --db Shop --collection product --type json --file /Users/agata/product.json
2020-04-04T14:01:48.496+0200    connected to: mongodb://localhost/
2020-04-04T14:01:48.529+0200    3 document(s) imported successfully. 0 document(s) failed to import.
MacBook-Pro-Agata:~ agata$ mongoimport --db Shop --collection purchase --type json --file /Users/agata/purchase.json
2020-04-04T14:02:38.797+0200    connected to: mongodb://localhost/
2020-04-04T14:02:38.833+0200    3 document(s) imported successfully. 0 document(s) failed to import.
MacBook-Pro-Agata:~ agata$
```

Opis: Kolekcja customer składa się z pól prostych i jednego pola złożonego (tj. adres), które są niezbędne do dokonania zakupu online i otrzymania zamówionego towaru

Opis: Kolekcja product składa się jedynie z pól prostych, które zawierają podstawowe informacje o danym produkcie, pozwalające na jego szybką identyfikację, uzyskania ceny wybranego przez klienta produktu oraz sprawdzenia czy wybrany przedmiot jest jeszcze dostępny

Opis: Kolekcja purchase zawiera pole customer_id, dzięki któremu z łatwością identyfikujemy odpowiedniego klienta, posiada tablicę products, gdzie znajdują się zakupione przedmioty wraz z id danego produktu, liczbą sztuk oraz ewentualną zniżką, jeżeli ten produkt jest takową objęty . Znajdujące się tu pole data zawiera istotną informację w przypadku zwrotów, a także jeśli chodzi o termin dostarczenia przesyłki (np. max tydzień od daty zakupu)