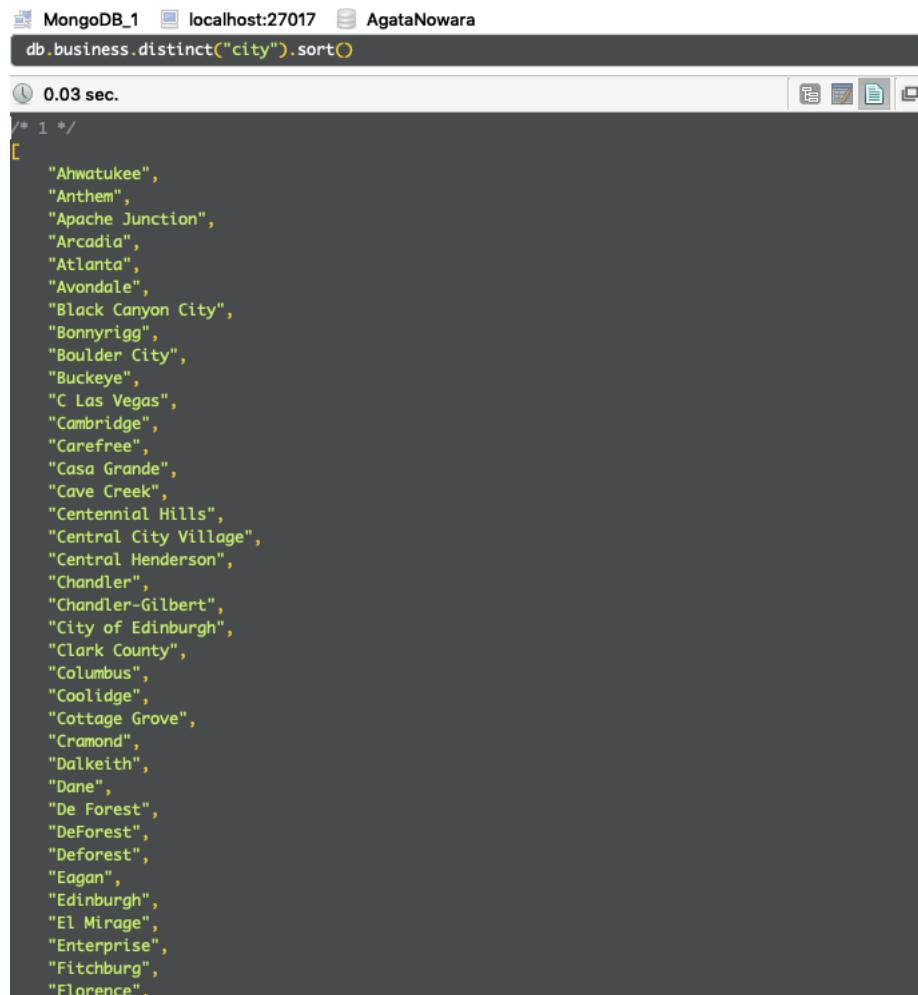


## Zad 1

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

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

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



The screenshot shows the MongoDB shell interface. The title bar reads "MongoDB\_1 localhost:27017 AgataNowara". The command line shows "db.business.distinct("city").sort()". Below the command, the output is displayed: "0.03 sec." followed by a list of city names. The list starts with "Ahwatukee", ends with "Florence", and includes many other names like "Anthem", "Apache Junction", "Arcadia", "Atlanta", "Avondale", etc. The list is sorted alphabetically.

```
/* 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",
  "Deforest",
  "Eagan",
  "Edinburgh",
  "El Mirage",
  "Enterprise",
  "Fitchburg",
  "Florence",
]
```

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

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



The screenshot shows the MongoDB shell interface. The title bar reads "MongoDB\_1 localhost:27017 AgataNowara". The command line shows "db.review.find({date: {\$gte: "2011-01-01"}}).count()". Below the command, the output is displayed: "0.68 sec." followed by the count value "880318".

```
db.review.find({date: {$gte: "2011-01-01"}}).count()
880318
```

- 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})
```

```
/* 1 */
{
  "_id" : ObjectId("5e79dfa4457b40a09cd322b"),
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
  "name" : "Charter Communications",
  "stars" : 1.5
}

/* 2 */
{
  "_id" : ObjectId("5e79dfa4457b40a09cd3239"),
  "full_address" : "6401 University Ave\nMiddleton, WI 53562",
  "name" : "Crandalls Carryout & Catering",
  "stars" : 4.0
}

/* 3 */
{
  "_id" : ObjectId("5e79dfa4457b40a09cd323f"),
  "full_address" : "6230 University Ave\nMiddleton, WI 53562",
  "name" : "Mi Cocina",
  "stars" : 3.0
}

/* 4 */
{
  "_id" : ObjectId("5e79dfa4457b40a09cd326e"),
  "full_address" : "1901 Cayuga St\nMiddleton, WI 53562",
  "name" : "Soup Factory",
  "stars" : 3.0
}

/* 5 */
{
  "_id" : ObjectId("5e79dfa4457b40a09cd3289"),
  "full_address" : "1632 W Main St\nSun Prairie, WI 53590",
  "name" : "Deli Roma",
  "stars" : 4.0
}
```

- 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})
```

```
/* 1 */
{
  "_id" : ObjectId("5e79e17b870abad36eef611d"),
  "yelping_since" : "2009-08",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "Bernard",
  "user_id" : "xp3SPgFgnW2vc5Zj5uV8SEA",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : [],
  "elite" : []
}
```

```

/* 2 */
{
    "_id" : ObjectId("5e79e17d870abad36ef05f41"),
    "yelping_since" : "2011-12",
    "votes" : {
        "funny" : 0,
        "useful" : 2,
        "cool" : 2
    },
    "review_count" : 10,
    "name" : "'Anastacia",
    "user_id" : "qJLc0rYtazeVBiTPPtfSA",
    "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_WSDcEg",
    "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}}
])

```

_id	count
InLd0PcxrMzLo6-j9VE0eg	1.0
J4NY4AsCVPPFn_QzJPgp-g	1.0
UtVMTQwKxQMJBQS1buEqCw	1.0
7dnH0aZnfc2HxSbJk8cdZw	1.0
Zou7W24BGZQULHEq1vhUUh	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"}}
])

```

The screenshot shows the MongoDB Compass interface with a query results window. The title bar says 'MongoDB\_1 localhost:27017 AgataNowara'. The results pane displays the following data:

```

/* 1 */
{
  "name" : "Koller True Value Hardware",
  "count" : 1.0
}

/* 2 */
{
  "name" : "The O2 Bar & Spa",
  "count" : 1.0
}

/* 3 */
{
  "name" : "Crowne Plaza Phoenix Airport",
  "count" : 1.0
}

/* 4 */
{
  "name" : "Paradise Bakery And Cafe",
  "count" : 1.0
}

/* 5 */
{
  "name" : "Multrees Walk",
  "count" : 1.0
}

/* 6 */
{
  "name" : "AZ TMJ",
  "count" : 1.0
}

```

- 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.

```

/* 1 */
{
  "_id" : "aAfKYj5DacC91NqPpU4iow",
  "averageGrade" : 4.0
}

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

/* 3 */
{
  "_id" : "cc9KFNrcY9gA7t9D1a3FpA",
  "averageGrade" : 4.5272727272727273
}

/* 4 */
{
  "_id" : "wn7GiuQQ_0VrPXNc1wVTDA",
  "averageGrade" : 4.333333333333333
}

/* 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.

```

/* 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" : 4.6666666666666667
}

/* 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()
```

The screenshot shows the MongoDB shell interface. The command `db.business.find({stars:2.0}).count()` is entered, followed by its execution time (0.034 sec.) and the count (1576). The results pane is empty at the bottom.

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

The screenshot shows the MongoDB shell interface. The command `db.business.remove({stars:2.0})` is entered, followed by its execution time (0.079 sec.) and the message "Removed 1576 record(s) in 80ms". The results pane is empty at the bottom.

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

The screenshot shows the MongoDB shell interface. The command `db.business.find({stars:2.0}).count()` is entered again, followed by its execution time (0.031 sec.) and the result (0). The results pane is empty at the bottom.

## 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"})
```

The screenshot shows the MongoDB Compass interface. At the top, there's a code editor window with the following JavaScript code:

```
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"})
}
```

Below the code editor, a status bar indicates "0.001 sec." and "Inserted 1 record(s) in 1ms".

At the bottom, another code editor window shows the inserted document:

```
/* 1 */
{
  "_id" : ObjectId("5e860dacc62f58ac469c6faa"),
  "votes" : {
    "funny" : 0.0,
    "useful" : 0.0,
    "cool" : 0.0
  },
  "user_id" : "my_id",
  "review_id" : "some_review",
  "stars" : 4.5,
  "date" : ISODate("2020-04-02T16:07:08.042Z"),
  "text" : "I really recommend this place",
  "type" : "review",
  "business_id" : "some_business"
}
```

### 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")
```

The screenshot shows the MongoDB Compass interface. At the top, there's a code editor window with the following JavaScript code:

```
function findBusinessesFromCategory(category)
{
  return db.business.find({categories: {$regex: category}})
}
findBusinessesFromCategory("Event")
```

```

findBusinessesFromCategory("Event")
business 0.004 sec.

/* 1 */
{
    "_id" : ObjectId("5e79dfa04457b40a09cd3231"),
    "business_id" : "XrzTVrJajs0yLvhFI9vkQ",
    "full_address" : "5508 Broadhead St\nMc Farland, WI 53558",
    "hours" : {},
    "open" : true,
    "categories" : [
        "Hotels & Travel",
        "Bed & Breakfast",
        "Event Planning & Services",
        "Hotels"
    ],
    "city" : "Mc Farland",
    "review_count" : 9,
    "name" : "Parsonage Bed & Breakfast the",
    "neighborhoods" : [],
    "longitude" : -89.285206,
    "state" : "WI",
    "stars" : 5.0,
    "latitude" : 43.015931,
    "attributes" : {
        "Accepts Credit Cards" : true,
        "Wi-Fi" : "free",
        "Price Range" : 2
    },
    "type" : "business"
}

/* 2 */
{
    "_id" : ObjectId("5e79dfa04457b40a09cd3239"),
    "business_id" : "HxPpZSY6Q1eARuiahhra6A",
    "full_address" : "6401 University Ave\nMiddleton, WI 53562",
    "hours" : {},
    "open" : false,
    "categories" : [
        "Event Planning & Services",
        "Party & Event Planning",
        "Caterers"
    ],
    "city" : "Middleton",
    "review_count" : 5,
    "name" : "Crandalls Carryout & Catering",
    "neighborhoods" : [],
    "longitude" : -89.4918,
    "state" : "WI",
    "stars" : 4.0,
    "latitude" : 43.093265,
    "attributes" : {}
}

```

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("MWhR9Lv0dRbqtu1I_DRFBg", "Monitor")

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

```

The screenshot shows the MongoDB Compass interface with two panes. The left pane displays a query log for a session named 'AgataNowara' on 'localhost:27017'. The query renames a user from 'Monitor' to 'Matthew'. The right pane shows a detailed view of a single document from the 'user' collection, which contains fields like '\_id', 'yelping\_since', 'votes', 'review\_count', 'name', 'user\_id', 'friends', 'fans', 'average\_stars', 'type', 'compliments', and 'elite'.

```

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("MWhR9Lv0dRbqtu1I_DRFBg", "Monitor")
db.user.find({name:"Monitor"})

```

0.149 sec.  
Updated 1 existing record(s) in 149ms

```

user 0.133 sec.
/* 1 */
{
  "_id" : ObjectId("5e79e17a870abad36eeeea107"),
  "yelping_since" : "2011-12",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "Monitor",
  "user_id" : "MWhR9Lv0dRbqtu1I_DRFBg",
  "friends" : [
    "8Y2EN4XNNhnwssuPb31sJg",
    "A1jPleJ99kXZ3t9wQ3np-g",
    "resYiOoGkQg6q0agtj_1GA",
    "skl10nkjMqD4GdFpVhU88Q",
    "3Ss2aqrSo07WbxE2GcLlfQ",
    "SGy3JDbhtzDTTB07unqQxg",
    "Aez4Y1F0m2ucIfmfzPZfjw",
    "ft4jx_9cRWyKFR0xv_MIrw",
    "QmZOAYM7ITbTdkTn6ay_ag",
    "_l18_te9lifeArhNdCffy1g",
    "H6ovBilqI3cSunx27XbPUg",
    "A8pmiEiiWjXci_oeigLNZQ",
    "fdSfg45S-P-FcVF-VIU0ew"
  ],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : {},
  "elite" : false
}

```

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(SortsAscending("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("$business_id", Accumulators.sum("count", 1)),
        Aggregates.sort(SortsAscending("count")),
        Aggregates.Lookup("business", "_id", "business_id", "business_data"),
        Aggregates.unwind("$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());
    }
}
```

```
119● public static void main(String[] args) throws UnknownHostException {  
120     MongoZad mongoZad = new MongoZad();  
121     mongoZad.distinctCitiesWithBusinesses();  
122 }  
  
Console  
<terminated> MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (3 kwie 2020, 14:25:41)  
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  
Deforest  
Eagan  
Edinburgh  
El Mirage  
Enterprise  
Fitchburg  
Florence  
Fort Kinnaird  
Fort McDowell
```

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;
}
```

```
119  public static void main(String[] args) throws UnknownHostException {
120      MongoZad mongoZad = new MongoZad();
121      mongoZad.countReviewsAfter2011();
122  }
123
124
125
126 }

Console 
<terminated> MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (3 kwie 2020, 14:35)
kw 03, 2020 2:35:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[127.0.0.1:27017], mode=SIMPLE, requiredClusterType=UNSPECIFIED}
kw 03, 2020 2:35:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
kw 03, 2020 2:35:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:418}] to 127.0.0.1:27017
kw 03, 2020 2:35:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=127.0.0.1:27017, type=STANDALONE, state=CONNECTED}
kw 03, 2020 2:35:16 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:419}] to 127.0.0.1:27017
880318
```

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());
    }
}
```

```
119  public static void main(String[] args) throws UnknownHostException {
120      MongoZad mongoZad = new MongoZad();
121      mongoZad.findClosedBusinesses();
122  }
123
124
125
126 }

Console 
<terminated> MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (3 kwie 2020, 14:36)
Las Vegas, NV 89139, name=The Taste of Chicago, stars=3.0}
Document{{_id=5e79dfac4457b40a09cdb487, full_address=1749 W Main St
Mesa, AZ 85201, name=Urban Heat, stars=3.5}
Document{{_id=5e79dfac4457b40a09cdb495, full_address=3102 N 16th St
Phoenix, AZ 85016, name=Da Burger Shack, stars=4.0}
Document{{_id=5e79dfac4457b40a09cdb497, full_address=7343 S 89th Pl
Mesa, AZ 85212, name=Fence Busters, stars=5.0}
Document{{_id=5e79dfac4457b40a09cdb499, full_address=8810 S Maryland Pkwy
Southeast
Las Vegas, NV 89123, name=Don Antonio's Pizzeria, stars=3.5}
Document{{_id=5e79dfac4457b40a09cdb4a4, full_address=255 King Street N
Waterloo, ON N2J 2Y8, name=Ausanda Chocolate, stars=4.5}
Document{{_id=5e79dfac4457b40a09cdb4c5, full_address=100 Portobello High Street
Edinburgh EH15 1AL, name=Annie & Belle, stars=4.5}
Document{{_id=5e79dfac4457b40a09cdb4ca, full_address=154 N Country Club Dr
Mesa, AZ 85201, name=5 Star BBQ & Grill, stars=4.0}
Document{{_id=5e79dfac4457b40a09cdb4cf, full_address=122 State St
Capitol
Madison, WI 53703, name=The Baker's Window, stars=4.5}
Document{{_id=5e79dfac4457b40a09cdb4dc, full_address=1639 W Warm Springs Rd
Henderson, NV 89014, name=Mediterranean Grill, stars=3.0}
Document{{_id=5e79dfac4457b40a09cdb4e4, full_address=5115 Spring Mountain Rd
Ste 103
Chinatown
Las Vegas, NV 89146, name=Kung Fu Chef, stars=3.5}
Document{{_id=5e79dfac4457b40a09cdb4ea, full_address=Gold Canyon, AZ 85118, name=Firehouse Bar &
Document{{_id=5e79dfac4457b40a09cdb4eb, full_address=3306 University Ave
Madison, WI 53705, name=Szechuan, stars=3.0}
Document{{_id=5e79dfac4457b40a09cdb4fe, full_address=1945 E Indian School Rd
Phoenix, AZ 85016, name=Grotown Horticulture Supply, stars=3.5}
```

- 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());
    }
}

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

- 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 }

MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (3 kw 2020, 14:42:55)
Document{{count=1, business_data=Document{{name=Kolar Tru Value Hardware}}}}
Document{{count=1, business_data=Document{{name=The O2 Bar & Spa}}}}
Document{{count=1, business_data=Document{{name=Crown Plaza Phoenix Airport}}}}
Document{{count=1, business_data=Document{{name=Paradise Bakery And Cafe}}}}
Document{{count=1, business_data=Document{{name=Mutrees Walk}}}}
Document{{count=1, business_data=Document{{name=AZ TM}}}}
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=Monahan 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=Zechuan}}}}
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 }

MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (3 kw 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=4.6666666666666667, 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=4.4324324324325, 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=4.3333333333333, 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=4.6666666666666667, business_data=Document{{name=Huger Mercy Living Center}}}}
Document{{avgStars=4.384615384615385, business_data=Document{{name=Peoria Cafe}}}}
Document{{avgStars=4.3333333333333333, 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=4.6666666666666667, business_data=Document{{name=Taste & Sounds of Soul}}}}
Document{{avgStars=4.25, business_data=Document{{name=Billet Bar}}}}
Document{{avgStars=4.45, business_data=Document{{name=Subway}}}}
Document{{avgStars=4.45, business_data=Document{{name=Dunkin' Donuts}}}}
Document{{avgStars=4.3000000000000003, business_data=Document{{name=Arby's}}}}

```

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;
}

128● public void remove2StarsBusinesses()
129 {   db.getCollection("businesses").deleteMany(Filters.eq("stars", 2));
130 }
131
132● public long count2StarsBusinesses()
133 {   MongoCollection<Document> business = db.getCollection("business");
134     long counter = business.countDocuments(Filters.eq("stars", 2));
135     return counter;
136 }
137
138● public static void main(String[] args) throws UnknownHostException {
139     MongoZad mongoZad = new MongoZad();
140     mongoZad.remove2StarsBusinesses();
141     System.out.println(mongoZad.count2StarsBusinesses());
142 }
```

The screenshot shows a Java application running in an IDE. The code implements methods to remove documents from the 'businesses' collection where the 'stars' field is 2.0 and to count such documents. It also includes a main method to demonstrate the functionality. The execution output in the console shows the application connecting to a MongoDB instance at 127.0.0.1:27017 and performing the specified operations.

```
<terminated> MongoZad [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (3 kwi 2020, 15:10:26 - 15:10:29)
Kw1 03, 2020 3:10:28 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[127.0.0.1:27017], mode=SIMPLE, requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}
Kw1 03, 2020 3:10:28 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
Kw1 03, 2020 3:10:28 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:438}] to 127.0.0.1:27017
Kw1 03, 2020 3:10:28 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=127.0.0.1:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionNumber:3, minWireVersion:0, maxWireVersion:0}, maxBsonObjectSize=2147483647, maxMessageSize=4194304, maxWriteBatchSize=16384, roundTripTime=0}
Kw1 03, 2020 3:10:28 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:439}] to 127.0.0.1:27017
0
```

## 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": "Nadarem", "number": 7, "postal_code": "30-308"}, "e-mail": "js@onet.pl", "phone": "603405671"}
```

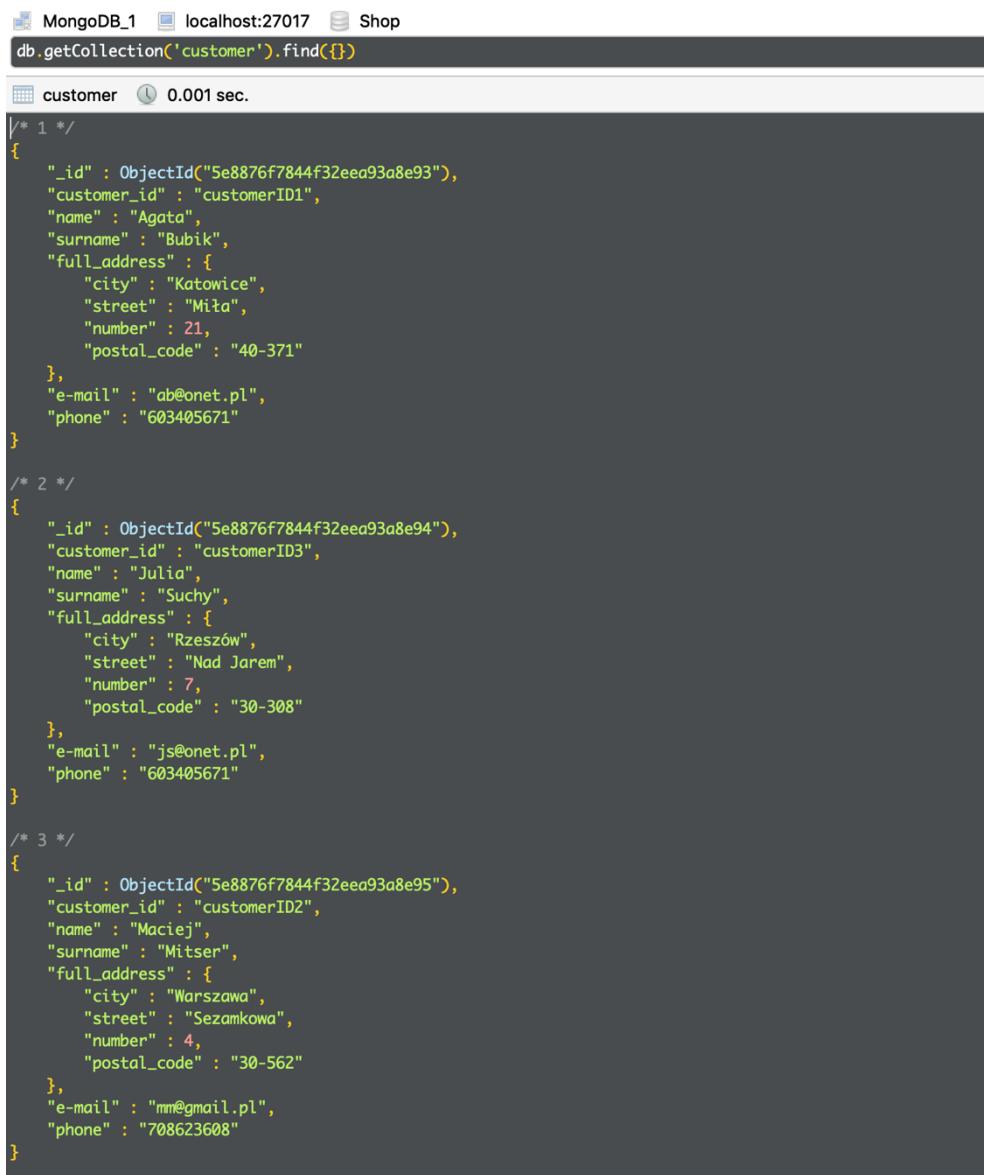
```
Users > agata > {} purchase.json > ...
1  {"order_id": "orderID1", "customer_id": "customerID1", "date": "2020-04-01", "products": [{"product_id": "productID1", "quantity": 1, "discount": 0.1}], "order_id": "orderID2", "customer_id": "customerID2", "date": "2020-04-02", "products": [{"product_id": "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}
```

Zainportowanie 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



The screenshot shows the MongoDB Compass interface with the database 'Shop' selected. The 'customer' collection is open, displaying three documents. Each document contains fields: \_id, customer\_id, name, surname, and full\_address. The full\_address field is an object containing city, street, number, and postal\_code.

```
/* 1 */
{
  "_id" : ObjectId("5e8876f7844f32eea93a8e93"),
  "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 */
{
  "_id" : ObjectId("5e8876f7844f32eea93a8e94"),
  "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"
}

/* 3 */
{
  "_id" : ObjectId("5e8876f7844f32eea93a8e95"),
  "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"
}
```

**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)

```
MongoDB_1   localhost:27017   Shop
db.getCollection('product').find({})

product  0.002 sec.

/* 1 */
{
  "_id" : ObjectId("5e88772c3bd1b0c9e93a68d4"),
  "product_id" : "productID1",
  "category" : "bag",
  "brand" : "Gucci",
  "name" : "bag1",
  "price" : 1279,
  "quantity_in_stock" : 20
}

/* 2 */
{
  "_id" : ObjectId("5e88772c3bd1b0c9e93a68d5"),
  "product_id" : "productID3",
  "category" : "sport equipment",
  "brand" : "Gregory",
  "name" : "backpack1",
  "price" : 699,
  "quantity_in_stock" : 30
}

/* 3 */
{
  "_id" : ObjectId("5e88772c3bd1b0c9e93a68d6"),
  "product_id" : "productID2",
  "category" : "sunglasses",
  "brand" : "Rayban",
  "name" : "ClubMaster",
  "price" : 699,
  "quantity_in_stock" : 40
}
```

```
MongoDB_1   localhost:27017   Shop
db.getCollection('purchase').find({})

purchase  0.001 sec.

/* 1 */
{
  "_id" : ObjectId("5e88775e45c19a4330665e4f"),
  "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
    }
  ]
}

/* 2 */
{
  "_id" : ObjectId("5e88775e45c19a4330665e50"),
  "order_id" : "orderID1",
  "customer_id" : "customerID1",
  "date" : "2020-04-01",
  "products" : [
    {
      "product_id" : "productID1",
      "quantity" : 1,
      "discount" : 0.1
    }
  ]
}

/* 3 */
{
  "_id" : ObjectId("5e88775e45c19a4330665e51"),
  "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
    }
  ]
}
```