

Bazy danych – MongoDB – zadanie

Imię i nazwisko: Bartosz Kordek

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

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

New Connection localhost:27017 BartoszKordek2

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

business 0.12 sec.

Key	Value	Type
▼ (1) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Ahwatukee	String
▼ (2) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Anthem	String
▼ (3) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Apache Junction	String
▼ (4) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Arcadia	String
▼ (5) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Atlanta	String
▼ (6) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Avondale	String
▼ (7) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Black Canyon City	String
▼ (8) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Bonnyrigg	String
▼ (9) { 1 field }	{ 1 field }	Object
▼ { 1 field }	{ 1 field }	Object
city	Boulder City	String
▼ (10) { 1 field }	{ 1 field }	Object

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

Pierwszy sposób - z wykorzystaniem funkcji aggregate, mapowaniem do tablicy, następnie wykorzystanie funkcji length

```
New Connection localhost:27017 BartoszKordek2
db.getCollection('review').aggregate([
  { $match: {"date": {$gte: "2011-01-01"}} }
]).toArray().length
9.97 sec.
880319
```

Drugi sposób z wykorzystaniem funkcji find, a następnie wykorzystanie funkcji count do zliczania rekordów.

```
New Connection localhost:27017 BartoszKordek2
db.getCollection('review').find({"date": {$gte: "2011-01-01"}}).count()
1.46 sec.
880319
```

Jak widać powyżej, wykorzystując drugi sposób szybciej można uzyskać wyniki.

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

```
New Connection localhost:27017 BartoszKordek2
db.getCollection('business').aggregate([
  { $match: {"open": false}},
  { $project : { name: "$name" , address:"$full_address", stars:"$stars" } }
])
business 0.006 sec.
```

Wyniki:

business 0.006 sec.		
Key	Value	Type
▼ (1) ObjectId("5fa67584d02e7129ab9c6d22")	{ 4 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d22")	ObjectId
name	Charter Communications	String
address	4156 County Rd B Mc Farland, WI 53558	String
stars	1.5	Double
▼ (2) ObjectId("5fa67584d02e7129ab9c6d2f")	{ 4 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d2f")	ObjectId
name	Crandalls Carryout & Catering	String
address	6401 University Ave Middleton, WI 53562	String
stars	4.0	Double
▼ (3) ObjectId("5fa67584d02e7129ab9c6d36")	{ 4 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d36")	ObjectId
name	Mi Cocina	String
address	6230 University Ave Middleton, WI 53562	String
stars	3.0	Double
▼ (4) ObjectId("5fa67584d02e7129ab9c6d5c")	{ 4 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d5c")	ObjectId
name	Stamm House At Pheasant Branch	String
address	6625 Century Ave Middleton, WI 53562	String
stars	2.0	Double
▼ (5) ObjectId("5fa67584d02e7129ab9c6d61")	{ 4 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d61")	ObjectId
name	Tangles	String
address	6661 University Ave Ste 103 Middleton, WI 53562	String
stars	2.0	Double
▼ (6) ObjectId("5fa67584d02e7129ab9c6d6f")	{ 4 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d6f")	ObjectId
name	Soup Factory	String
address	1901 Campus St Middleton, WI 53562	String

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

```
New Connection localhost:27017 BartoszKordek2
db.getCollection('user').aggregate([
  { $match:
    { $and: [
      { "votes.funny": 0 },
      { "votes.useful": 0 }
    ] } },
  { $sort: { name: 1 } }
])

user 0.49 sec.
```

user0.49 sec.

Key

Value

(1) ObjectId("5fa6760cc64caa9e2093a137")

{ 12 fields }

_id

ObjectId("5fa6760cc64caa9e2093a137")

yelping_since

2009-08

votes

{ 3 fields }

review_count

1

name

Bernard

user_id

xP3SPgfgW2vc5Zj5uV8SEA

friends

[0 elements]

fans

0

average_stars

5.0

type

user

compliments

{ 0 fields }

elite

[0 elements]

(2) ObjectId("5fa67611c64caa9e20960b4d")

{ 12 fields }

_id

ObjectId("5fa67611c64caa9e20960b4d")

yelping_since

2013-03

votes

{ 3 fields }

review_count

1

name

,Maria

user_id

Os5f3TNpM7_A8IDNEdPX2g

friends

[0 elements]

fans

0

average_stars

5.0

type

user

compliments

{ 0 fields }

View Document

```
{
  "_id" : ObjectId("5fa6760cc64caa9e2093a137"),
  "yelping_since" : "2009-08",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : " Bernard",
  "user_id" : "xP3SPgfgW2vc5Zj5uV8SEA",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}
```

Bernard jest pierwszy, ponieważ jak widać w dokumencie jest spacja na początku nazwy użytkownika (" Bernard"). Poniżej lepiej widać, że wyniki są posortowane alfabetycznie.

Key	Value	Type
name	Abbey	String
user_id	JETHr3SL6h7q_qL5uvAdlw	String
friends	[0 elements]	Array
fans	0	Int32
average_stars	5.0	Double
type	user	String
compliments	{ 0 fields }	Object
elite	[0 elements]	Array
(17) ObjectId("5fa67611c64caa9e2095d1af")	{ 12 fields }	Object
_id	ObjectId("5fa67611c64caa9e2095d1af")	ObjectId
yelping_since	2013-05	String
votes	{ 3 fields }	Object
review_count	2	Int32
name	Abbe	String
user_id	T3nNMh4MfM-yLTNEweRvKA	String
friends	[0 elements]	Array
fans	0	Int32
average_stars	5.0	Double
type	user	String
compliments	{ 0 fields }	Object
elite	[0 elements]	Array
(18) ObjectId("5fa6760dc64caa9e209456da")	{ 12 fields }	Object
_id	ObjectId("5fa6760dc64caa9e209456da")	ObjectId
yelping_since	2014-05	String
votes	{ 3 fields }	Object
review_count	2	Int32
name	Abbey	String
user_id	hH4BfwVO_JB9TTqrQCGnKQ	String
friends	[0 elements]	Array
fans	0	Int32
average_stars	5.0	Double

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

New Connection localhost:27017 BartoszKordek2

```
db.getCollection('tip').aggregate([
  { $match:
    { $and: [
      {"date": { $gte: "2012-01-01"}},
      {"date": { $lte: "2012-12-31"}}
    ]}
  },
  { $group: { _id: { business: "$business_id"}, total: { $sum: 1 } } },
  { $sort: { total: -1 } }
])
```

tip 1.33 sec.

Key	Value	Type
▼ (1) { 1 field }	{ 2 fields }	Object
▼ (1) _id	{ 1 field }	Object
business	jf67Z1pnwEIRSXllpQHilg	String
total	1084.0	Double
▼ (2) { 1 field }	{ 2 fields }	Object
▼ (2) _id	{ 1 field }	Object
business	hW0Ne_HTHEAgGF1rAdmR-g	String
total	622.0	Double
▼ (3) { 1 field }	{ 2 fields }	Object
▼ (3) _id	{ 1 field }	Object
business	2e2e7WgqU1BnpxmQL5jbfw	String
total	430.0	Double
▼ (4) { 1 field }	{ 2 fields }	Object
▼ (4) _id	{ 1 field }	Object
business	CsNOg-u_wCuXSt9Z-xU92Q	String
total	374.0	Double
▼ (5) { 1 field }	{ 2 fields }	Object
▼ (5) _id	{ 1 field }	Object
business	AtjsjFzaIWqJ7S9DUFQ4bw	String
total	351.0	Double
▼ (6) { 1 field }	{ 2 fields }	Object
▼ (6) _id	{ 1 field }	Object
business	zt1TpTwJ6y9n551sw9TaEg	String

Wyniki posortowałem malejąco wg liczby napiwków.

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

New Connection localhost:27017 BartoszKordek2

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

review 2.46 sec.

review 2.46 sec.		
Key	Value	Type
▼ (1) { 1 field }	{ 2 fields }	Object
> (1) _id	{ 1 field }	Object
avg	4.7	Double
▼ (2) { 1 field }	{ 2 fields }	Object
> (2) _id	{ 1 field }	Object
avg	5.0	Double
▼ (3) { 1 field }	{ 2 fields }	Object
> (3) _id	{ 1 field }	Object
avg	4.2	Double
▼ (4) { 1 field }	{ 2 fields }	Object
> (4) _id	{ 1 field }	Object
avg	4.666666666666667	Double
▼ (5) { 1 field }	{ 2 fields }	Object
> (5) _id	{ 1 field }	Object
avg	4.2109375	Double
▼ (6) { 1 field }	{ 2 fields }	Object
> (6) _id	{ 1 field }	Object
avg	4.6363636363636364	Double
▼ (7) { 1 field }	{ 2 fields }	Object
> (7) _id	{ 1 field }	Object
avg	4.72093023255814	Double
▼ (8) { 1 field }	{ 2 fields }	Object
> (8) _id	{ 1 field }	Object
avg	4.083333333333333	Double
▼ (9) { 1 field }	{ 2 fields }	Object
> (9) _id	{ 1 field }	Object
avg	4.9	Double
▼ (10) { 1 field }	{ 2 fields }	Object
> (10) _id	{ 1 field }	Object

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

Do celów zadania zduplikowałem kolekcję business.

```
New Connection localhost:27017 BartoszKordek2
db.getCollection('business_copy').find({}).count()
0.001 sec.
42153
```

Jak widać powyżej liczba elementów w kolekcji to 42.153.

```
New Connection localhost:27017 BartoszKordek2
db.getCollection('business_copy').find({ stars : 2.0 }).count()
0.024 sec.
1576
```

Znajduje się w niej 1.576 elementów posiadających pole stars równe 2.0.

New Connection localhost:27017 BartoszKordek2

```
db.business_copy.deleteMany({ stars : 2.0 })
```

0.051 sec.

Key	Value	Type
(1)	{ 2 fields }	Object
acknowledged	true	Boolean
deletedCount	1576.0	Double

Widać, że zostało usuniętych 1.576 elementów.

New Connection localhost:27017 BartoszKordek2

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

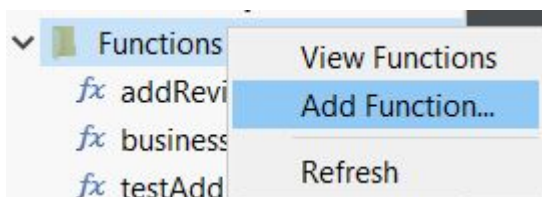
0.023 sec.

Fetches 0 record(s) in 23ms

W kolekcji business_copy nie znajdują się już elementy, które posiadają 2 gwiazdki.

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

Z powodu błędu Robo 3T, funkcja może nie zapisać się w standardowy sposób.



Konieczne jest wówczas wykorzystanie funkcji db.system.js.save().

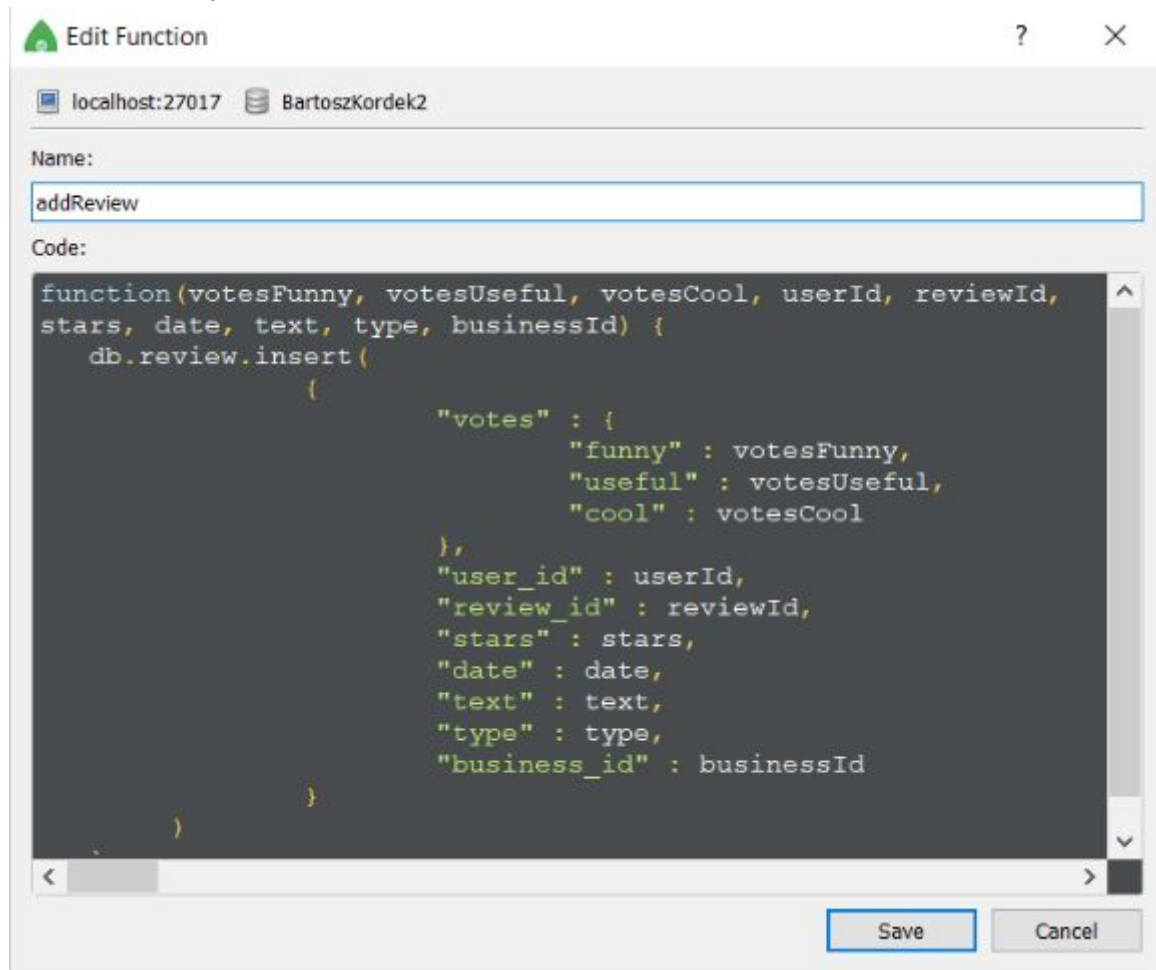
New Connection localhost:27017 BartoszKordek2

```
db.system.js.save(
{
  _id : "addReview",
  value: function addNewReview(votesFunny, votesUseful, votesCool, userId, reviewId, stars, date, text, type, businessId) {
    db.review.insert(
      {
        "votes" : {
          "funny" : votesFunny,
          "useful" : votesUseful,
          "cool" : votesCool
        },
        "user_id" : userId,
        "review_id" : reviewId,
        "stars" : stars,
        "date" : date,
        "text" : text,
        "type" : type,
        "business_id" : businessId
      }
    )
  }
})
```

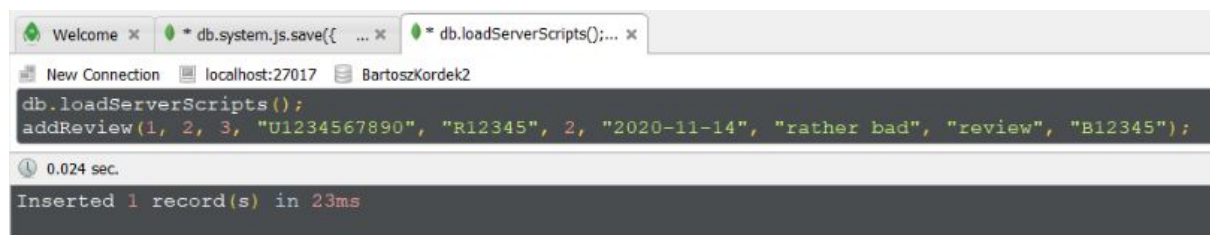
Poniżej cała funkcja (całość nie zmieściła się w konsoli Robo 3T).

```
1 function addNewReview(votesFunny, votesUseful, votesCool, userId, reviewId, stars, date, text, type, businessId) {
2   db.review.insert(
3     {
4       "votes" : {
5         "funny" : votesFunny,
6         "useful" : votesUseful,
7         "cool" : votesCool
8       },
9       "user_id" : userId,
10      "review_id" : reviewId,
11      "stars" : stars,
12      "date" : date,
13      "text" : text,
14      "type" : type,
15      "business_id" : businessId
16    }
17  )
18 }
```

Widać, że funkcja została dodana.



Przykładowe wywołanie funkcji:



Rekord został dodany do bazy danych.

The screenshot shows the MongoDB interface. At the top, a query is entered: `db.getCollection('review').find({"user_id" : "U1234567890"})`. Below the query, a table displays the result. The table has two columns: 'Key' and 'Value'. The first row shows a key: `> (1) ObjectId("5fb038ab3d950d36508197de")` and a value: `{ 9 fields }`. Below the table, there is a 'View Document' section. It shows the document structure for the 'review' collection. The document is a JSON object with the following fields: `["_id" : ObjectId("5fb038ab3d950d36508197de"), "votes" : { "funny" : 1.0, "useful" : 2.0, "cool" : 3.0 }, "user_id" : "U1234567890", "review_id" : "R12345", "stars" : 2.0, "date" : "2020-11-14", "text" : "rather bad", "type" : "review", "business_id" : "B12345"]`.

Key	Value
<code>> (1) ObjectId("5fb038ab3d950d36508197de")</code>	<code>{ 9 fields }</code>

```
{
  "_id" : ObjectId("5fb038ab3d950d36508197de"),
  "votes" : {
    "funny" : 1.0,
    "useful" : 2.0,
    "cool" : 3.0
  },
  "user_id" : "U1234567890",
  "review_id" : "R12345",
  "stars" : 2.0,
  "date" : "2020-11-14",
  "text" : "rather bad",
  "type" : "review",
  "business_id" : "B12345"
}
```

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.

Utworzenie funkcji.

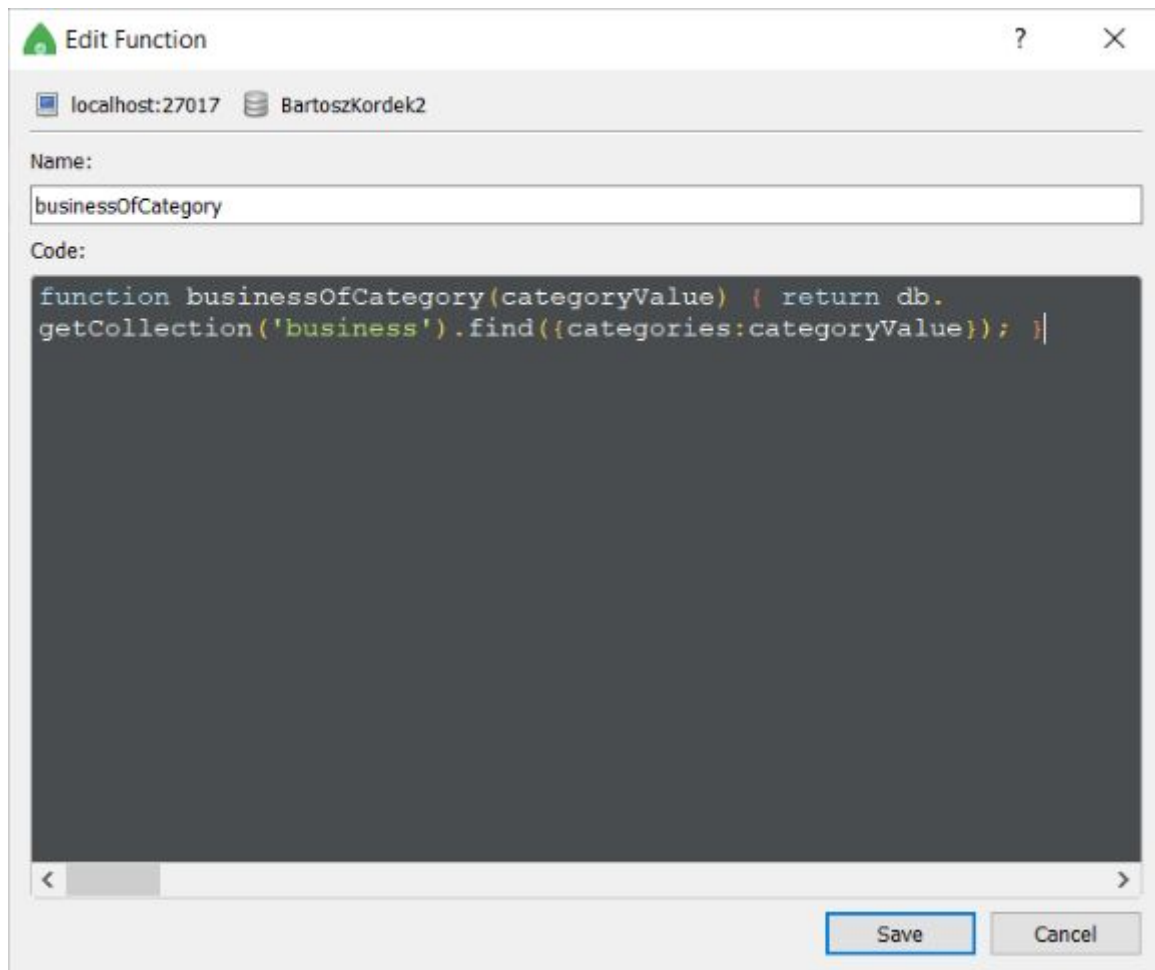
The screenshot shows the MongoDB interface. A query is entered: `db.system.js.save({ _id: "businessOfCategory", value : function businessOfCategory(categoryValue) { return db.getCollection('business').find({categories:categoryValue}); } })`. Below the query, the execution time is shown: `0.002 sec.`. At the bottom, a status message indicates: `Updated 1 new record(s) in 2ms`.

```
db.system.js.save(
{
  _id: "businessOfCategory",
  value : function businessOfCategory(categoryValue) { return db.getCollection('business').find({categories:categoryValue}); }
})
```

0.002 sec.

Updated 1 new record(s) in 2ms

Funkcja została utworzona.



Wywołanie funkcji.



Wynik wywołania funkcji:

Key	Value	Type
▼ (1) ObjectId("5fa67584d02e7129ab9c6d24")	{ 16 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d24")	ObjectId
business_id	RgDg-k9S5YD_BaxMckifkg	String
full_address	631 S Main St De Forest, WI 53532	String
> hours	{ 7 fields }	Object
open	true	Boolean
▼ categories	[2 elements]	Array
[0]	Chinese	String
[1]	Restaurants	String
city	De Forest	String
review_count	3	Int32
name	Chang Jiang Chinese Kitchen	String
> neighborhoods	[0 elements]	Array
longitude	-89.3437217	Double
state	WI	String
stars	4.0	Double
latitude	43.2408748	Double
> attributes	{ 4 fields }	Object
type	business	String
▼ (2) ObjectId("5fa67584d02e7129ab9c6d27")	{ 16 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d27")	ObjectId
business_id	uGykseHzyS5xAMWoN6YUqA	String
full_address	505 W North St De Forest, WI 53532	String
> hours	{ 7 fields }	Object
open	true	Boolean
▼ categories	[2 elements]	Array
[0]	American (Traditional)	String
[1]	Restaurants	String
city	De Forest	String

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

Utworzenie funkcji.

```
New Connection localhost:27017 BartoszKordek2

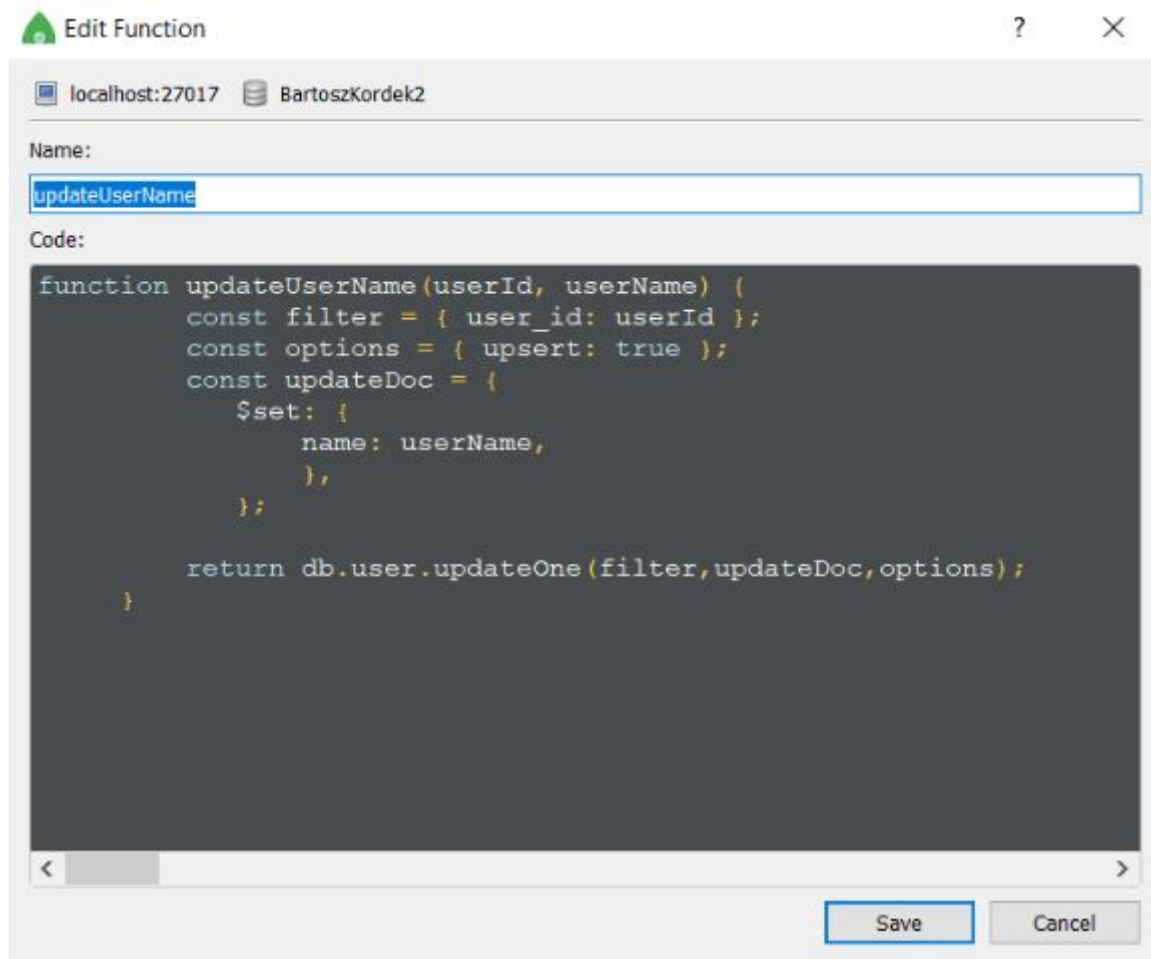
db.system.js.save(
  {
    _id: "updateUserName",
    value: function updateUserName(userId, userName) {
      const filter = { user_id: userId };
      const options = { upsert: true };
      const updateDoc = {
        $set: {
          name: userName,
        },
      };

      return db.user.updateOne(filter, updateDoc, options);
    }
  }
);

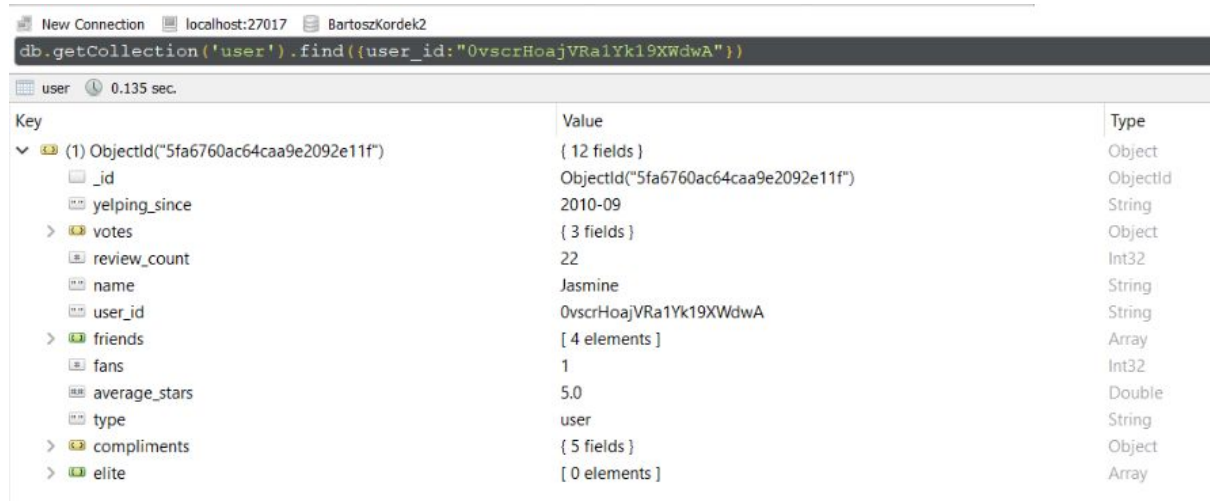
0.001 sec.

Updated 1 new record(s) in 2ms
```

Funkcja została utworzona.



Wybór użytkownika do zmiany nazwy.



Wywołanie funkcji.

New Connection localhost:27017 BartoszKordek2		
<pre>db.loadServerScripts(); var userId = "0vscrHoajVRa1Yk19XWdwA"; var userName = "Jessica"; updateUserName(userId, userName);</pre>		
0.004 sec.		
Key	Value	Type
▼ (1)	{ 3 fields }	Object
acknowledged	true	Boolean
matchedCount	1.0	Double
modifiedCount	1.0	Double

Sprawdzenie, że nazwa użytkownika została zmieniona.

New Connection localhost:27017 BartoszKordek2		
<pre>db.getCollection('user').find({user_id:"0vscrHoajVRa1Yk19XWdwA"})</pre>		
user 0.135 sec.		
Key	Value	Type
▼ (1) ObjectId("5fa6760ac64caa9e2092e11f")	{ 12 fields }	Object
_id	ObjectId("5fa6760ac64caa9e2092e11f")	ObjectId
yelping_since	2010-09	String
> votes	{ 3 fields }	Object
review_count	22	Int32
name	Jessica	String
user_id	0vscrHoajVRa1Yk19XWdwA	String
> friends	[4 elements]	Array
fans	1	Int32
average_stars	5.0	Double
type	user	String
> compliments	{ 5 fields }	Object
> elite	[0 elements]	Array

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

Dokładnie nie zrozumiałem o co chodzi w treści zadania, więc przesyłam dwa rozwiązania zadań, w jaki sposób można było to zrozumieć.

1) Średnia wartość polubień z napiwków dla każdego z biznesów:

New Connection localhost:27017 BartoszKordek2		
<pre>db.tip.mapReduce(function(){ emit(this.business_id, this.likes); }, function(key, values){ return Array.avg(values); }, { query: {type: "tip"}, out: "businessLikesAvg" })</pre>		
10.4 sec.		

10.4 sec.		
Key	Value	Type
▼ (1)	{ 6 fields }	Object
result	businessLikesAvg	String
ok	1.0	Double
▼ _o	{ 2 fields }	Object
result	businessLikesAvg	String
ok	1.0	Double
> _keys	[2 elements]	Array
▼ _db	{ 3 fields }	Object
> _mongo	{ 5 fields }	Object
_name	BartoszKordek2	String
_session	{ 28 fields }	Object
▼ _coll	{ 4 fields }	Object
> _mongo	{ 5 fields }	Object
> _db	{ 3 fields }	Object
_shortName	businessLikesAvg	String
_fullName	BartoszKordek2.businessLikesAvg	String

Została utworzona nowa kolekcja businessLikesAvg, w której znajdują się biznesy wraz ze średnią ilością polubień w napiwkach.

New Connection localhost:27017 BartoszKordek2		
db.getCollection('businessLikesAvg').find({})		
businessLikesAvg 0.006 sec.		
Key	Value	Type
▼ (1) iOvBH_NczhWWvsNP4vmy-A	{ 2 fields }	Object
_id	iOvBH_NczhWWvsNP4vmy-A	String
value	0.0	Double
▼ (2) tD_DOWa2DitPuENvzaO0-Q	{ 2 fields }	Object
_id	tD_DOWa2DitPuENvzaO0-Q	String
value	0.0	Double
▼ (3) ABED8k4aldmpePfU-032lw	{ 2 fields }	Object
_id	ABED8k4aldmpePfU-032lw	String
value	0.0	Double
▼ (4) cEe7zcX3REFZsyCDmGqRqQ	{ 2 fields }	Object
_id	cEe7zcX3REFZsyCDmGqRqQ	String
value	0.0	Double
▼ (5) SfHdMvLXrOeJmgpQB6AFRA	{ 2 fields }	Object
_id	SfHdMvLXrOeJmgpQB6AFRA	String
value	0.0	Double
▼ (6) i0W4TZcoRbOYeRfDfQEX8g	{ 2 fields }	Object
_id	i0W4TZcoRbOYeRfDfQEX8g	String
value	0.0	Double
▼ (7) UU6Dzrd04yfpW4OSacno0A	{ 2 fields }	Object
_id	UU6Dzrd04yfpW4OSacno0A	String
value	0.0	Double
▼ (8) 9eOg-2aKtyoWxoTKYeNHJA	{ 2 fields }	Object

Możemy sprawdzić, które biznesy mają średnią średnią ilość polubień z napiwków większą niż 0.

New Connection localhost:27017 BartoszKordek2

```
db.getCollection('businessLikesAvg').find({value: {"$gt": 0}})
```

businessLikesAvg 0.005 sec.

Key	Value	Type
▼ (1) ZHvCePeFeZ9PN41JA-ZzQ	{ 2 fields }	Object
_id	ZHvCePeFeZ9PN41JA-ZzQ	String
value	0.0408163265306122	Double
▼ (2) Q_Rwcv9APBLggk2KAsBEvg	{ 2 fields }	Object
_id	Q_Rwcv9APBLggk2KAsBEvg	String
value	0.006944444444444444	Double
▼ (3) EI_0LAdJ1R0-nL_IHzBT-g	{ 2 fields }	Object
_id	EI_0LAdJ1R0-nL_IHzBT-g	String
value	0.0689655172413793	Double
▼ (4) ITbhfASs9pyUcoHQN7xKVA	{ 2 fields }	Object
_id	ITbhfASs9pyUcoHQN7xKVA	String
value	0.25	Double
▼ (5) 4wDCGshJmq2CpZEImXX6QQ	{ 2 fields }	Object
_id	4wDCGshJmq2CpZEImXX6QQ	String
value	0.05555555555555556	Double
▼ (6) G9LwkTSdKOyOGqd5BbulkA	{ 2 fields }	Object
_id	G9LwkTSdKOyOGqd5BbulkA	String
value	0.0123456790123457	Double
▼ (7) NdQMBelvh0TTe1plrnFYmA	{ 2 fields }	Object
_id	NdQMBelvh0TTe1plrnFYmA	String
value	0.02083333333333333	Double
▼ (8) DO3Gk17RyJVW7zYMCtYPnw	{ 2 fields }	Object
_id	DO3Gk17RyJVW7zYMCtYPnw	String
value	0.00819672131147541	Double
▼ (9) eKOLoTJ7X8C4ZeqrThYwfA	{ 2 fields }	Object
_id	eKOLoTJ7X8C4ZeqrThYwfA	String
value	0.193548387096774	Double
▼ (10) 54_mej5zTsyf9KoohdA3hg	{ 2 fields }	Object
_id	54_mej5zTsyf9KoohdA3hg	String
value	0.5	Double

2) Średnia ilość napiwków ze wszystkich biznesów

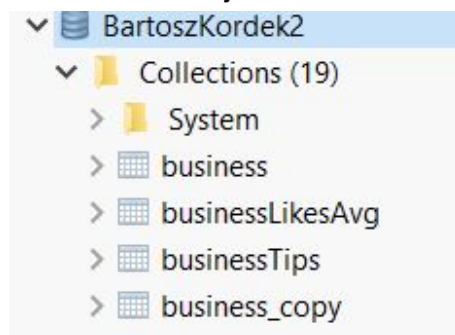
Najpierw tworzę funkcję przy wykorzystaniu map reduce obliczającą sumę wszystkich napiwków dla każdego biznesu.

New Connection localhost:27017 BartoszKordek2

```
db.tip.mapReduce(  
  function(){  
    emit(this.business_id, 1);  
  },  
  function(key, value){  
    return Array.sum(value);  
  },  
  {  
    query: {type: 'tip'},  
    out: "businessTips"  
  }  
)
```

10 sec.		
Key	Value	Type
▼ (1)	{ 6 fields }	Object
result	businessTips	String
ok	1.0	Double
▼ _o	{ 2 fields }	Object
result	businessTips	String
ok	1.0	Double
▼ _keys	[2 elements]	Array
[0]	result	String
[1]	ok	String
▼ _db	{ 3 fields }	Object
> _mongo	{ 5 fields }	Object
_name	BartoszKordek2	String
> _session	{ 28 fields }	Object
▼ _coll	{ 4 fields }	Object
> _mongo	{ 5 fields }	Object
> _db	{ 3 fields }	Object
_shortName	businessTips	String
_fullName	BartoszKordek2.businessTips	String

Jak widać kolekcja została utworzona



New Connection localhost:27017 BartoszKordek2		
db.getCollection('businessTips').find({})		
businessTips 0.002 sec.		
Key	Value	Type
▼ (1) EW8rqAt1czCzdKi8g9P5dQ	{ 2 fields }	Object
_id	EW8rqAt1czCzdKi8g9P5dQ	String
value	10.0	Double
▼ (2) 4rFs17eUaCuB7um1VFwQWw	{ 2 fields }	Object
_id	4rFs17eUaCuB7um1VFwQWw	String
value	4.0	Double
▼ (3) bwpkUfLhpZfECGO1pCbwljg	{ 2 fields }	Object
_id	bwpkUfLhpZfECGO1pCbwljg	String
value	4.0	Double
▼ (4) hEVQ4Zumf0QLKJXBpx2qEQ	{ 2 fields }	Object
_id	hEVQ4Zumf0QLKJXBpx2qEQ	String
value	21.0	Double
▼ (5) 63aWlQlZFevezwDwGPph8Q	{ 2 fields }	Object
_id	63aWlQlZFevezwDwGPph8Q	String
value	5.0	Double
▼ (6) hg1bivzMP0Z3c9eb5FhaRA	{ 2 fields }	Object
_id	hg1bivzMP0Z3c9eb5FhaRA	String
value	1.0	Double
▼ (7) cQMfAN2YVUcJn8wK2M_wGw	{ 2 fields }	Object
_id	cQMfAN2YVUcJn8wK2M_wGw	String
value	55.0	Double
▼ (8) X8tfJzFU-pNxWnZk1la03Q	{ 2 fields }	Object
_id	X8tfJzFU-pNxWnZk1la03Q	String
value	11.0	Double

Wynik:

New Connection localhost:27017 BartoszKordek2		
db.businessTips.aggregate([{\$group: {_id: ObjectId(), avgTips: {\$avg: "\$value"}}}])		
businessTips 0.102 sec.		
Key	Value	Type
▼ (1) ObjectId("5fce6fadf60cc2447e83500c")	{ 2 fields }	Object
_id	ObjectId("5fce6fadf60cc2447e83500c")	ObjectId
avgTips	13.4812263867063	Double

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

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

```
private AggregateIterable<Document> businessCities() {  
  
    Document city = new Document("city", "$city");  
    Document groupField = new Document("_id", city);  
    Document group = new Document("$group", groupField);  
    Document sort = new Document("$sort", new Document("_id", 1));  
  
    MongoCollection<Document> collection = mdb.getCollection("business");  
    List<Document> pipeline = Arrays.asList(group, sort);  
    return collection.aggregate(pipeline);  
  
}
```

Wyniki:

```
64 public static void main(String args[]) throws UnknownHostException {
65     MongoHomework mongoHomework = new MongoHomework();
66
67     //ZAD 6 a
68     AggregateIterable<Document> businessCities = mongoHomework.businessCities();
69     for(Document d : businessCities) {
70         System.out.println(d.toJson());
71     }

```

<terminated> MongoHomework (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (29 lis 2020, 23:02:03)

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

Wyniki pokrywają się z tymi uzyskanymi przy pomocy narzędzia Robo 3T.

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

```
private int moviesAfter2011() {
    int counter = 0;
    Document gte2011 = new Document("$gte", "2011-01-01");
    Document date = new Document("date", gte2011);
    Document match = new Document("$match", date);

    MongoCollection<Document> collection = mdb.getCollection("review");
    List<Document> pipeline = Arrays.asList(match);
    AggregateIterable<Document> results = collection.aggregate(pipeline);

    for(Document d : results) counter++;

    return counter;
}
```

Można zadanie wykonać innym (szybszym sposobem):

```
private int moviesAfter2011BetterSolution() {
    Document gte2011 = new Document("$gte", "2011-01-01");
   DBObject query = new BasicDBObject();
    query.put("date", gte2011);
    DBCollection gettedCollection = db.getCollection("review");
    return gettedCollection.find(query).count();
}
```

Wyniki:

```
153 //ZAD 6 b
154 startTime = System.nanoTime();
155 System.out.print("FIRST SOLUTION: Result: "+mongoHomework.moviesAfter2011());
156 endTime = System.nanoTime();
157 System.out.println(" Execution Time: "+(endTime-startTime)+" [ns]");
158 startTime = System.nanoTime();
159 System.out.print("FASTER SOLUTION: Result: "+mongoHomework.moviesAfter2011BetterSolution());
160 endTime = System.nanoTime();
161 System.out.println(" Execution Time: "+(endTime-startTime)+" [ns]");
162
```

<terminated> MongoHomework (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (30 lis 2020, 20:13:16)

lis 30, 2020 8:13:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[127.0.0.1:27017], mode=SINGLE, requiredClusterType=UNKNOWN, ser
lis 30, 2020 8:13:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:15}] to 127.0.0.1:27017
lis 30, 2020 8:13:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=127.0.0.1:
lis 30, 2020 8:13:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
lis 30, 2020 8:13:18 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:16}] to 127.0.0.1:27017
FIRST SOLUTION: Result: 880319 Execution Time: 11904080200 [ns]
FASTER SOLUTION: Result: 880319 Execution Time: 1429624700 [ns]

Jak widać, wyniki są takie same jak w przypadku wykorzystania narzędzia Robo 3T.
Rozwiązanie drugie jest ponad 8x szybsze od pierwszego.

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

```
private AggregateIterable<Document> getClosedBusinesses(){  
  
    Document match = new Document("$match", new Document("open", true));  
    Document elements = new Document("name", "$name");  
    elements.put("address", "$full_address");  
    elements.put("stars", "$stars");  
    Document project = new Document("$project", elements);  
  
    MongoCollection<Document> collection = mdb.getCollection("business");  
    List<Document> pipeline = Arrays.asList(match, project);  
    return collection.aggregate(pipeline);  
}
```

```
104  
105 //ZAD 6 c  
106 AggregateIterable<Document> closedBusinesses = mongoHomework.getClosedBusinesses();  
107 for(Document d : closedBusinesses) {  
108     System.out.println(d.toJson());  
109 }  
110  
***  
  
Console Problems Javadoc Declaration Coverage  
<terminated> MongoHomework (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (29 lis 2020, 23:30:40)  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fbf"}, "name": "The Dirty Band", "address": "Las Vegas, NV", "stars": 5.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fbf"}, "name": "Connie Lopez - Wardley Real Estate", "address": "7670 W. Lake Mead Blvd\nSte 100\nSummerlin\nLas Vegas, NV 89128", "stars": 4.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fbf"}, "name": "Bruce E Crowley, DDS", "address": "9510 W Sahara Ave\nWestside\nLas Vegas, NV 89117", "stars": 4.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc1"}, "name": "Swirls", "address": "6501 N Greenway Pkwy\nPhoenix, AZ 85254", "stars": 4.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc2"}, "name": "Leather Like New LLC", "address": "3202 S 40th St\nSte 12\nPhoenix, AZ 85040", "stars": 3.5 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc3"}, "name": "Krayvings", "address": "11770 W Charleston Blvd\nSte 150\nSummerlin\nLas Vegas, NV 89135", "stars": 4.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc4"}, "name": "Santos Lucha Libre", "address": "9822 N 7th St\nSte 7\nPhoenix, AZ 85020", "stars": 5.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc5"}, "name": "DMV Drop", "address": "217 N Stephanie Blvd\nHenderson, NV 89074", "stars": 5.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc6"}, "name": "The Metro Plaza Hotel", "address": "10220 N Metro Pkwy E\nPhoenix, AZ 85051", "stars": 3.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc7"}, "name": "Yuki Shaved Snow", "address": "8414 Farm Rd\nSte 150\nCentennial\nLas Vegas, NV 89131", "stars": 4.5 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc8"}, "name": "Zipps Sports Grill", "address": "690 S Mill Ave\nBldg C, Ste 103\nTempe, AZ 85281", "stars": 3.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fc9"}, "name": "Innovative Dentistry", "address": "7260 S Cimarron Rd\nSte 150\nSouthwest\nLas Vegas, NV 89113", "stars": 5.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fca"}, "name": "Taqueria El Chino", "address": "1803 W Van Buren St\nPhoenix, AZ 85007", "stars": 5.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fcb"}, "name": "iCracked Professional iPhone, iPod and iPad Repair", "address": "Phoenix, AZ 85027", "stars": 4.0 }  
{ "_id": {"$oid": "5fa67586d02e7129ab9d0fcc"}, "name": "Thousand Nights Hookah Lounge", "address": "2050 N Alma School Rd\nChandler, AZ 85225", "stars": 4.5 }
```

Wyniki pokrywają się z tymi uzyskanymi przy pomocy narzędzia Robo 3T.

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

```
private AggregateIterable<Document> getNoFunnyAndUsefulUsers(){

    Document votesFunny = new Document("votes.funny", 0);
    Document votesUseful = new Document("votes.useful", 0);
    Document match = new Document("$match", new Document("$and", Arrays.asList(votesFunny,votesUseful)));
    Document sort = new Document("$sort", new Document("name", 1));

    MongoCollection<Document> collection = mdb.getCollection("user");
    List<Document> pipeline = Arrays.asList(match, sort);
    return collection.aggregate(pipeline);
}
```

Wyniki:

```
//ZAD 6 d
AggregateIterable<Document> noFunnyAndUsefulUsers = mongoHomework.getNoFunnyAndUsefulUsers();
for(Document d : noFunnyAndUsefulUsers) {
    System.out.println(d.toJson());
}
```

<terminated> MongoHomework (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (29 lis 2020, 23:34:35)

```
{ "_id": {"$oid": "5fa6760ac64caa9e209301ed"}, "yelping_since": "2009-09", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "kevin", "user_id": "M3-f"},
{"_id": {"$oid": "5fa6760dc64caa9e209407fa"}, "yelping_since": "2009-10", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 4, "name": "kevin", "user_id": "Upul"},
{"_id": {"$oid": "5fa6760fc64caa9e20951001"}, "yelping_since": "2011-11", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 3, "name": "kevin", "user_id": "wxdq"},
{"_id": {"$oid": "5fa67611c64caa9e2095fd57"}, "yelping_since": "2007-07", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "kevin", "user_id": "kYv"},
{"_id": {"$oid": "5fa67612c64caa9e20967595"}, "yelping_since": "2011-10", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 3, "name": "kevin", "user_id": "3LJ"},
{"_id": {"$oid": "5fa67612c64caa9e20967cd1"}, "yelping_since": "2011-07", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "kiki", "user_id": "HSnst"},
{"_id": {"$oid": "5fa6760ec64caa9e2093d00d"}, "yelping_since": "2009-04", "votes": {"funny": 0, "useful": 0, "cool": 1}, "review_count": 2, "name": "kikyochan", "user_id": "i"},
{"_id": {"$oid": "5fa6760ec64caa9e2094c6e0"}, "yelping_since": "2011-07", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "kim", "user_id": "jMSjtf"},
{"_id": {"$oid": "5fa67612c64caa9e20967d97"}, "yelping_since": "2010-07", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "kimberly", "user_id": "t"},
{"_id": {"$oid": "5fa67613c64caa9e2096afcf"}, "yelping_since": "2005-11", "votes": {"funny": 0, "useful": 0, "cool": 1}, "review_count": 6, "name": "kip", "user_id": "1A70m"},
{"_id": {"$oid": "5fa67610c64caa9e20957017"}, "yelping_since": "2009-02", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "kirk", "user_id": "nTFX"},
{"_id": {"$oid": "5fa67611c64caa9e2095e841"}, "yelping_since": "2011-08", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 4, "name": "kJ", "user_id": "jFXV75"},
{"_id": {"$oid": "5fa67612c64caa9e20964156"}, "yelping_since": "2010-10", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 5, "name": "kosh", "user_id": "yM70"},
{"_id": {"$oid": "5fa67613c64caa9e20968c51"}, "yelping_since": "2008-11", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "kraig", "user_id": "gzVl"},
{"_id": {"$oid": "5fa67611c64caa9e2095de70"}, "yelping_since": "2008-07", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "kris", "user_id": "B7Gll"},
{"_id": {"$oid": "5fa6760cc64caa9e2093de47"}, "yelping_since": "2011-09", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "kristine", "user_id": "t"},
{"_id": {"$oid": "5fa6760ac64caa9e2092ea66"}, "yelping_since": "2011-04", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 3, "name": "krystal", "user_id": "et"},
{"_id": {"$oid": "5fa6760fc64caa9e20951afe"}, "yelping_since": "2010-03", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "krystal", "user_id": "4"},
{"_id": {"$oid": "5fa67613c64caa9e2096964e"}, "yelping_since": "2010-01", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "l", "user_id": "sxh-CC4"}
```

Wyniki pokrywają się z tymi uzyskanymi przy pomocy narzędzia Robo 3T.

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

```
private AggregateIterable<Document> getBusinessTips(){
    Document dateGreaterThanOrEqual = new Document("date", new Document("$gte", "2012-01-01"));
    Document dateLessThan = new Document("date", new Document("$lte", "2012-12-31"));
    Document match = new Document("$match", new Document("$and", Arrays.asList(dateGreaterThanOrEqual, dateLessThan)));

    Document businessTotal = new Document("_id", new Document("business", "$business_id"));
    businessTotal.put("total", new Document("$sum", 1));
    Document group = new Document("$group", businessTotal);
    Document sort = new Document("$sort", new Document("total", -1));

    List<Document> pipeline = Arrays.asList(match, group, sort);
    MongoCollection<Document> collection = mdb.getCollection("tip");
    return collection.aggregate(pipeline);
}
```

Wyniki:

```
147 //ZAD 6 e
148 AggregateIterable<Document> businessTips = mongoHomework.getBusinessTips();
149 for(Document d : businessTips) {
150     System.out.println(d.toJson());
151 }
152
153 }
154
155 }
156
```

Console Problems Javadoc Declaration Coverage

<terminated> MongoHomework (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (30 lis 2020, 00:37:00)

```
{ "_id": { "business": "cAe0tSTNBa9vfJc2K3ZgaQ", "total": 1 }
{ "_id": { "business": "avZwwAB-Vhs05eeXxgmu2Q", "total": 1 }
{ "_id": { "business": "SSMcyI7xx6wBi0VCcUxgaA", "total": 1 }
{ "_id": { "business": "sjW4oS0aXjwvJHli6ZPJzg", "total": 1 }
{ "_id": { "business": "PN6o2ktnGIn5HgF6a8XXBg", "total": 1 }
{ "_id": { "business": "RUcgIIvxzDbarVNzIUO-Bg", "total": 1 }
{ "_id": { "business": "6fgx1Tx-kTEUa0cA7181RQ", "total": 1 }
{ "_id": { "business": "d2AeNB4xw67E1Bzpo1Pm7Q", "total": 1 }
{ "_id": { "business": "2XFORMgw9kvUG_ZHm29-0A", "total": 1 }
{ "_id": { "business": "gy4MXrt_Ue48M_ScugtCvA", "total": 1 }
{ "_id": { "business": "4f3cvkJdYOFm-MLhSI02AA", "total": 1 }
{ "_id": { "business": "keYc-0lYxUV5ViKXY3lIag", "total": 1 }
{ "_id": { "business": "vSuRccwnSvTQ4KKfHwpiqQ", "total": 1 }
{ "_id": { "business": "s-jfTJnAIdyvIw2Fh2QcGg", "total": 1 }
{ "_id": { "business": "BKBsQ_xbwyzn6pDs_Ykbcg", "total": 1 }
{ "_id": { "business": "7_fkHRNNHsfVP0pQCWm4yQ", "total": 1 }
{ "_id": { "business": "2E1Kdwf4BXV8c7RC-ThkwA", "total": 1 }
{ "_id": { "business": "f9G8rYZ-HpQvekvizABeKg", "total": 1 }
{ "_id": { "business": "DENQD70TKne29w4KCSOq1g", "total": 1 }
{ "_id": { "business": "vxlyf1N5Uuy83wufcw1BEw", "total": 1 }
{ "_id": { "business": "BykY_dJiVk4CIY-XUTDapQ", "total": 1 }
{ "_id": { "business": "fFtDyJmVLPTDuQ61SgOxPw", "total": 1 }
{ "_id": { "business": "VfIIP3runJ9VDVsfnI6Lg", "total": 1 }
```

Wyniki pokrywają się z tymi uzyskanymi przy pomocy narzędzia Robo 3T.

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

```
private AggregateIterable<Document> getAvgBusinessStars(){
    Document businessStars = new Document("_id", new Document("business", "$business_id"));
    businessStars.put("avg", new Document("$avg", "$stars"));
    Document group = new Document("$group", businessStars);

    Document match = new Document("$match", new Document("avg", new Document("$gte", 4.0)));

    List<Document> pipeline = Arrays.asList(group, match);
    MongoCollection<Document> collection = mdb.getCollection("business");

    return collection.aggregate(pipeline);
}
```

Wyniki:

```
167 //ZAD 6 f
168 AggregateIterable<Document> avgBusinessStars = mongoHomework.getAvgBusinessStars();
169 for(Document d : avgBusinessStars) {
170     System.out.println(d.toJson());
171 }
172 }
173
```

Console Problems Javadoc Declaration Coverage

<terminated> MongoHomework (1) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (30 lis 2020, 00:50:48)

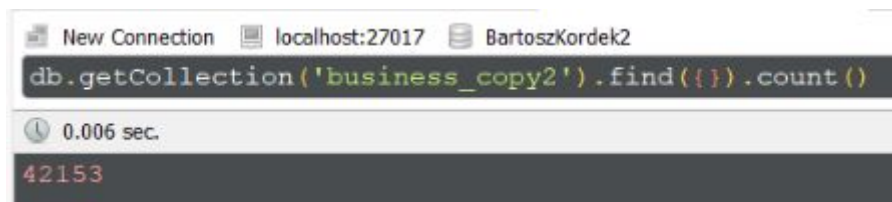
```
{"_id": {"business": "5nddZwZzT7pvC5vBjKRzoA"}, "avg": 4.0}
{"_id": {"business": "mTLifoWe5enFtQF_pvcnXQ"}, "avg": 4.0}
{"_id": {"business": "QDIXvG3jvpXP7vp2hQ3U6A"}, "avg": 5.0}
{"_id": {"business": "a66Y0mnzZN44iCL3Cu45DA"}, "avg": 5.0}
{"_id": {"business": "riRdxF4gyojFud08ME4FSQ"}, "avg": 4.5}
{"_id": {"business": "IfeS8KfF_oRQ15z8GW0iGA"}, "avg": 4.0}
{"_id": {"business": "8xDLKyrVVNLDaX9zzybZQQ"}, "avg": 4.0}
{"_id": {"business": "jTcLbRCOCnGB4wCgsIYswg"}, "avg": 4.5}
{"_id": {"business": "egEroiWI8XuPoXwnb7UkuQ"}, "avg": 4.0}
{"_id": {"business": "BZk5ij706Xgx8Yr_3EwEcQ"}, "avg": 4.5}
{"_id": {"business": "tqn7equmFCyrkkJ10iU1xA"}, "avg": 5.0}
{"_id": {"business": "3r5V8n1sabGkueQFsqoC6Q"}, "avg": 4.0}
{"_id": {"business": "aLcJrXy2ALwNwZyGJNx_9w"}, "avg": 4.0}
{"_id": {"business": "b_d4JmyOyOM8En7Vtjkbw"}, "avg": 4.0}
{"_id": {"business": "gWBxaHOMXuBPSY7CxLeL4Q"}, "avg": 4.5}
{"_id": {"business": "UgdRZzKwWlOukDQzhux7JQ"}, "avg": 5.0}
{"_id": {"business": "O_5RlNmIyBRzwGZAfwsZFA"}, "avg": 4.5}
{"_id": {"business": "uIC_xmGn5vReCGbf972x8g"}, "avg": 5.0}
{"_id": {"business": "5yb13Rm9doDsaAAIJwnsQw"}, "avg": 4.5}
{"_id": {"business": "1r9g60tGRMHkU4xYiLYdkw"}, "avg": 4.0}
{"_id": {"business": "TQWikKmbZEIZy90zq9yKtA"}, "avg": 4.5}
{"_id": {"business": "RqSkCwsq4lLyTEGiJRPnqA"}, "avg": 4.5}
{"_id": {"business": "aDp5WgyctZi8H66W-liAiw"}, "avg": 4.0}
{"_id": {"business": "DlsAjiaSaFc5I132MJFG_A"}, "avg": 4.0}
{"_id": {"business": "ywmgMT-ir0XdbVzCADppAg"}, "avg": 4.5}
```

Wyniki są identyczne jak przy wykorzystaniu narzędzia Robo 3T.

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

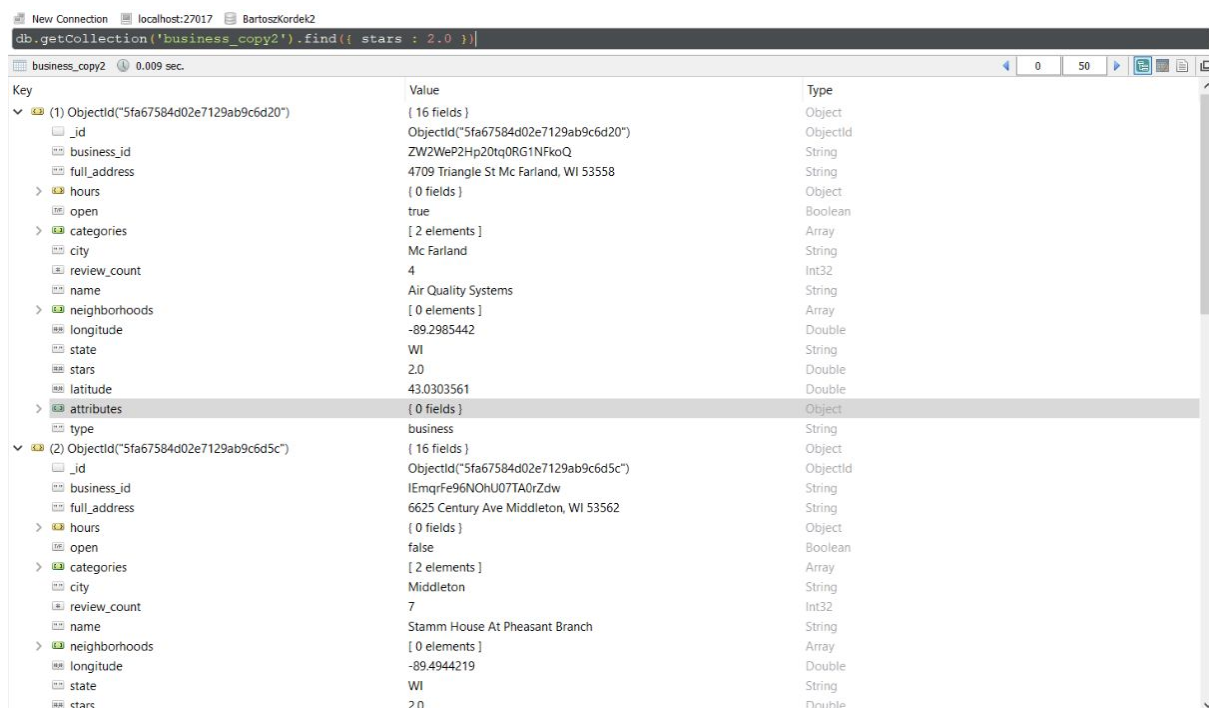
Do celów ćwiczenia skopiowałem (zduplikowałem) obecną kolekcję business i nazwałem ją business_copy2.

Liczba wszystkich dokumentów w kolekcji business_copy2 wynosi 42.153.



```
New Connection localhost:27017 BartoszKordek2
db.getCollection('business_copy2').find({}).count()
0.006 sec.
42153
```

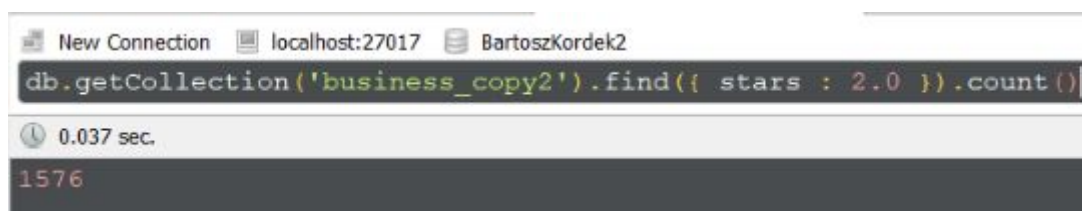
Możemy w niej zauważyć, że występują w niej firmy posiadające ocenę równą 2.0.



```
New Connection localhost:27017 BartoszKordek2
db.getCollection('business_copy2').find({ stars : 2.0 })
business_copy2 0.009 sec.
```

Key	Value	Type
(1) ObjectId("5fa67584d02e7129ab9c6d20")	{ 16 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d20")	ObjectId
business_id	ZW2WeP2Hp20tq0RG1NFkoQ	String
full_address	4709 Triangle St Mc Farland, WI 53558	String
hours	{ 0 fields }	Object
open	true	Boolean
categories	[2 elements]	Array
city	Mc Farland	String
review_count	4	Int32
name	Air Quality Systems	String
neighborhoods	[0 elements]	Array
longitude	-89.2985442	Double
state	WI	String
stars	2.0	Double
latitude	43.0303561	Double
attributes	{ 0 fields }	Object
type	business	String
(2) ObjectId("5fa67584d02e7129ab9c6d5c")	{ 16 fields }	Object
_id	ObjectId("5fa67584d02e7129ab9c6d5c")	ObjectId
business_id	lEmqrFe96NohU07TA0rZdw	String
full_address	6625 Century Ave Middleton, WI 53562	String
hours	{ 0 fields }	Object
open	false	Boolean
categories	[2 elements]	Array
city	Middleton	String
review_count	7	Int32
name	Stamm House At Pheasant Branch	String
neighborhoods	[0 elements]	Array
longitude	-89.4944219	Double
state	WI	String
stars	2.0	Double

Co więcej takich firm jest 1.576.



```
New Connection localhost:27017 BartoszKordek2
db.getCollection('business_copy2').find({ stars : 2.0 }).count()
0.037 sec.
1576
```

Funkcja:

```
private void remove2StarsBusinesses() {  
    MongoCollection<Document> collection = mdb.getCollection("business_copy2");  
    Document query = new Document("stars", 2.0);  
    collection.deleteMany(query);  
}
```

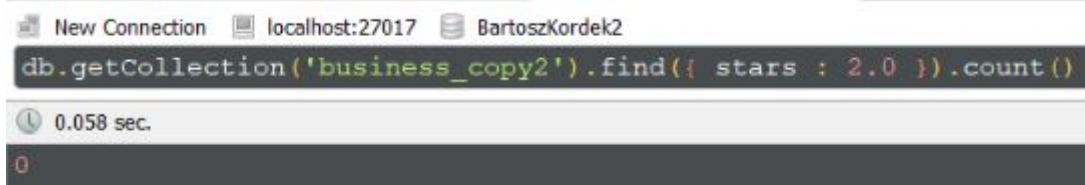
Wywołanie funkcji:

```
//ZAD 6 g  
mongoHomework.remove2StarsBusinesses();
```

Po wywołaniu funkcji można zauważyć, że ilość dokumentów zmniejszyła się i wynosi obecnie 40.577.



Po sprawdzeniu, można zauważyć, że nie występują już firmy posiadające 2 gwiazdki.



Co wskazuje, że wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0 zostały usunięte.

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.

Kolekcja Clients - reprezentująca klientów składać się będzie z następujących pól:

- `_id` - identyfikator dokumentu (klienta) [ObjectId]
- User Name - nazwa klienta [String] - ponieważ klient może tak naprawdę nie znać (a nawet nie powinien znać) Id dokumentu w bazie danych powiązanych z nim. Ponadto istnieje możliwość stworzenia nazwy jaką użytkownik będzie chciał, a nie jaką narzuci mu system. Pole powinno być unikatowe.
- First Name - imię [String]
- Last Name - nazwisko [String]
- Addresses - adresy - pole złożone określające adresy klienta. Czasami adres dostawy może się różnić od adresu podanego przy rejestracji. Składać się one będą z następujących pól:

- Address - adres [String]
- Postal Code - kod pocztowy [String]
- City - miasto [String]

Klient może posiadać kilka adresów. Dla celów, np. korespondencyjnych może być inny adres, lecz dla celów dostawy inny. Istnieje swoboda dodania adresu podstawowego i opcjonalnie adresu dostawy.

- Phone - numer telefonu [String] - na numerze telefonu nie będziemy prowadzić żadnych obliczeń, dlatego mimo, że występują same cyfry, zapiszemy go jako String. Ponadto mógłby być problem gdyby numer telefonu zaczynał się cyfrą "0" (np. jak jest w Wielkiej Brytanii).
- Email - adres email. Pole powinno być unikatowe.

Podane powyżej pola są niezbędne
Przykładowy dokument:

```
//Clients
{
  "_id" : ObjectId("5fc2cd2b33321682d52166d3"),
  "User Name": "johnsmith123",
  "First Name": "John",
  "Last Name": "Smith",
  "Addresses": {
    "Primary Address": {
      "Address": "7 Elm Street",
      "Postal Code": "22311",
      "City": "Los Angeles"
    },
    "Delivery Address": {
      "Address": "3368 Central Avenue",
      "Postal Code": "07662",
      "City": "Rochelle Park"
    }
  },
  "Phone": "8622526147",
  "Email": "johnsmith887@yahoo.com"
}
```

Kolekcja Products - reprezentująca produkty (przedmiot zakupu) składać się będzie z następujących pól:

- `_id` - identyfikator dokumentu (produktu) [ObjectId]
- Product Name - nazwa produktu [String]
- Description - opis produktu [String]
- Price - cena [NumberDecimal] - lepiej stosować niż double do reprezentacji ilości pieniędzy (można było również użyć integer i reprezentować wartość w groszach, centach, itp.
- Currency [String]
- Units in Stock [NumberInteger32] - wprowadzenie bez użycia NumberInt defaultowo ustawia typ na double, co w tym przypadku nie jest pożądane.
- Category - tablica Stringów - reprezentuje tzw. tagi wspomagające wyszukiwanie produktów

```
//Products
{
  "_id" : ObjectId("5fc5353e114ad21a2bc01315"),
  "Product Name": "Apple iPhone 11 64GB",
  "Description" : "The iPhone 11 was introduced in September 2019",
  "Price" : NumberDecimal("699.99"),
  "Currency": "USD",
  "Units in Stock": NumberInt(20),
  "Category" : [
    "Apple",
    "iPhone",
    "Mobile",
    "Premium",
    "64GB"
  ]
}
```

Kolekcja Orders - reprezentująca zamówienia (zakupy) składać się będzie z następujących pól:

- `_id` - identyfikator dokumentu (zamówienia) [ObjectId]
- ClientID - nr ID klienta z kolekcji Clients (id dokumentu) [String] - wartość z ObjectId, reprezentującego klienta
- ProductID - nr ID produktu z kolekcji Products (id dokumentu) [String] - wartość z ObjectId, reprezentującego produkt
- Units - ilość zamówionych sztuk [NumberInteger32]
- Dates - pole złożone reprezentujące daty charakterystyczne dla zamówienia, tj. data zamówienia, data płatności, data wysłania, data dostarczenia [ISODate] - data mogłaby być również przedstawiona jako String
- Shipper - przewoźnik [String]

```
//Orders
```

```
{
  "_id" : ObjectId("5fc2d25733321682d52166d5"),
  "ClientID": "5fc2cd2b33321682d52166d3",
  "ProductID": "5fc5353e114ad21a2bc01315",
  "Units": NumberInt(1),
  "Dates": {
    "Order": ISODate("2020-09-28T06:01:17.171Z"),
    "Payment": ISODate("2020-09-28T11:12:07.271Z"),
    "Shipped": ISODate("2020-10-01T08:00:01.271Z"),
    "Delivery": ISODate("2020-10-02T10:59:31.571Z")
  },
  "Shipper": "Fed Ex"
}
```

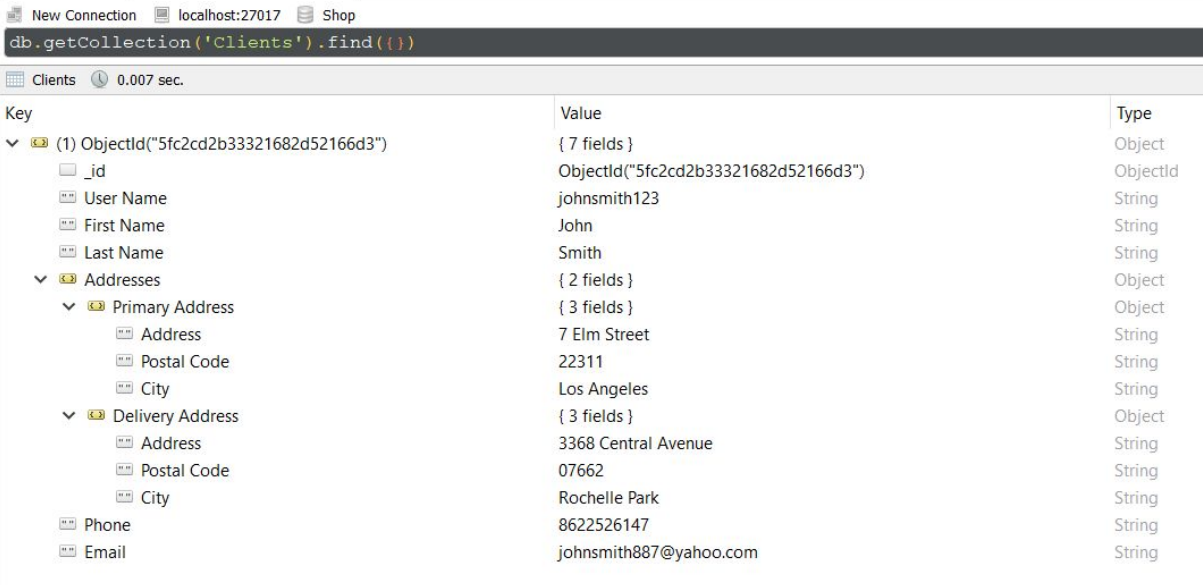
Wstawienie przykładowego klienta do bazy danych:

```
New Connection  localhost:27017  Shop
db.Clients.insert({
  "User Name": "johnsmith123",
  "First Name": "John",
  "Last Name": "Smith",
  "Addresses": {
    "Primary Address": {
      "Address": "7 Elm Street",
      "Postal Code": "22311",
      "City": "Los Angeles"
    },
    "Delivery Address": {
      "Address": "3368 Central Avenue",
      "Postal Code": "07662",
      "City": "Rochelle Park"
    }
  },
  "Phone": "8622526147",
})
```

0.004 sec.

Inserted 1 record(s) in 3ms

Można zobaczyć, że dokument z przykładowym klientem znajduje się w kolekcji Clients.

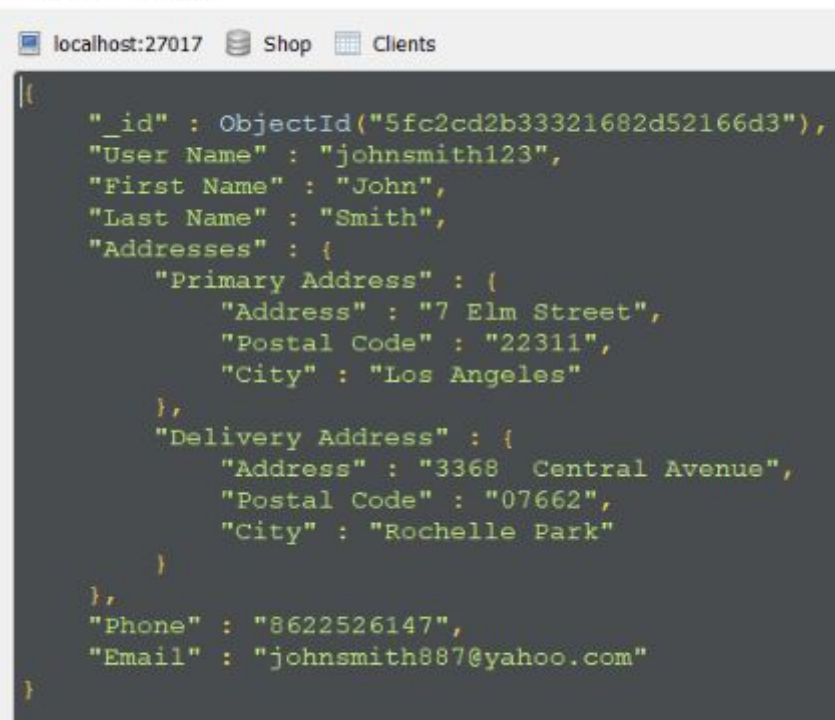


The screenshot shows the MongoDB Compass interface. At the top, the connection is 'localhost:27017' and the database is 'Shop'. The command bar contains `db.getCollection('Clients').find({})`. Below, the 'Clients' collection is selected, showing a document with a key of `(1) ObjectId("5fc2cd2b33321682d52166d3")`. The document structure is as follows:

Key	Value	Type
<code>(1) ObjectId("5fc2cd2b33321682d52166d3")</code>	<code>{ 7 fields }</code>	Object
<code>_id</code>	<code>ObjectId("5fc2cd2b33321682d52166d3")</code>	ObjectId
<code>User Name</code>	<code>johnsmith123</code>	String
<code>First Name</code>	<code>John</code>	String
<code>Last Name</code>	<code>Smith</code>	String
<code>Addresses</code>	<code>{ 2 fields }</code>	Object
<code>Primary Address</code>	<code>{ 3 fields }</code>	Object
<code>Address</code>	<code>7 Elm Street</code>	String
<code>Postal Code</code>	<code>22311</code>	String
<code>City</code>	<code>Los Angeles</code>	String
<code>Delivery Address</code>	<code>{ 3 fields }</code>	Object
<code>Address</code>	<code>3368 Central Avenue</code>	String
<code>Postal Code</code>	<code>07662</code>	String
<code>City</code>	<code>Rochelle Park</code>	String
<code>Phone</code>	<code>8622526147</code>	String
<code>Email</code>	<code>johnsmith887@yahoo.com</code>	String

A dokument w formacie JSON wygląda następująco:

 View Document



```
{
  "_id" : ObjectId("5fc2cd2b33321682d52166d3"),
  "User Name" : "johnsmith123",
  "First Name" : "John",
  "Last Name" : "Smith",
  "Addresses" : {
    "Primary Address" : {
      "Address" : "7 Elm Street",
      "Postal Code" : "22311",
      "City" : "Los Angeles"
    },
    "Delivery Address" : {
      "Address" : "3368 Central Avenue",
      "Postal Code" : "07662",
      "City" : "Rochelle Park"
    }
  },
  "Phone" : "8622526147",
  "Email" : "johnsmith887@yahoo.com"
}
```

Wstawianie przykładowego produktu do bazy danych:

```
New Connection  localhost:27017  Shop

db.Products.insert(
{
  "Product Name": "Apple iPhone 11 64GB",
  "Description" : "The iPhone 11 was introduced in September 2019, and Apple
  "Price" : NumberDecimal("699.99"),
  "Currency": "USD",
  "Units in Stock": NumberInt(20),
  "Category" : [
    "Apple",
    "iPhone",
    "Mobile",
    "Premium",
    "64GB"
  ]
}
)

0.004 sec.
Inserted 1 record(s) in 2ms
```

Można zobaczyć, że dokument z przykładowym produktem znajduje się w kolekcji Products.

```
New Connection  localhost:27017  Shop
db.getCollection('Products').find({})
```

Key	Value	Type
▼ (1) ObjectId("5fc5353e114ad21a2bc01315")	{ 7 fields }	Object
_id	ObjectId("5fc5353e114ad21a2bc01315")	ObjectId
Product Name	Apple iPhone 11 64GB	String
Description	The iPhone 11 was introduced in September 2019, and Apple has conti...	String
Price	699.99	Decimal128
Currency	USD	String
Units in Stock	20	Int32
▼ Category	[5 elements]	Array
[0]	Apple	String
[1]	iPhone	String
[2]	Mobile	String
[3]	Premium	String
[4]	64GB	String

Dokument w formacie JSON wygląda następująco:

```
View Document

localhost:27017  Shop  Products

{
  "_id" : ObjectId("5fc5353e114ad21a2bc01315"),
  "Product Name" : "Apple iPhone 11 64GB",
  "Description" : "The iPhone 11 was introduced in September 2019, and Apple has continued selling it at a reduced
starting price of $599 following the introduction of the iPhone 12 lineup. The iPhone 11 features a 6.1-inch display, a
dual-lens camera, and an A13 Bionic chip.",
  "Price" : NumberDecimal("699.99"),
  "Currency" : "USD",
  "Units in Stock" : 20,
  "Category" : [
    "Apple",
    "iPhone",
    "Mobile",
    "Premium",
    "64GB"
  ]
}
```

Wstawianie przykładowego dokumentu kupna do bazy:

```
New Connection  localhost:27017  Shop

db.Orders.insert(
  {
    "ClientID": "5fc2cd2b33321682d52166d3",
    "ProductID": "5fc5353e114ad21a2bc01315",
    "Units": NumberInt(1),
    "Dates": {
      "Order": ISODate("2020-09-28T06:01:17.171Z"),
      "Payment": ISODate("2020-09-28T11:12:07.271Z"),
      "Shipped": ISODate("2020-10-01T08:00:01.271Z"),
      "Delivery": ISODate("2020-10-02T10:59:31.571Z")
    },
    "Shipper": "Fed Ex"
  }
)

0.015 sec.

Inserted 1 record(s) in 14ms
```

Można zobaczyć, że dokument z przykładowym zakupem znajduje się w kolekcji Orders.

```
New Connection  localhost:27017  Shop
db.getCollection('Orders').find({})
```

Orders 0.004 sec.		
Key	Value	Type
▼ (1) ObjectId("5fc537dd114ad21a2bc01316")	{ 6 fields }	Object
_id	ObjectId("5fc537dd114ad21a2bc01316")	ObjectId
ClientID	5fc2cd2b33321682d52166d3	String
ProductID	5fc5353e114ad21a2bc01315	String
Units	1	Int32
▼ Dates	{ 4 fields }	Object
Order	2020-09-28 06:01:17.171Z	Date
Payment	2020-09-28 11:12:07.271Z	Date
Shipped	2020-10-01 08:00:01.271Z	Date
Delivery	2020-10-02 10:59:31.571Z	Date
Shipper	Fed Ex	String

Dokument w formacie JSON wygląda następująco:



View Document

```
localhost:27017 Shop Orders
{
  "_id" : ObjectId("5fc537dd114ad21a2bc01316"),
  "ClientID" : "5fc2cd2b33321682d52166d3",
  "ProductID" : "5fc5353e114ad21a2bc01315",
  "Units" : 1,
  "Dates" : {
    "Order" : ISODate("2020-09-28T06:01:17.171Z"),
    "Payment" : ISODate("2020-09-28T11:12:07.271Z"),
    "Shipped" : ISODate("2020-10-01T08:00:01.271Z"),
    "Delivery" : ISODate("2020-10-02T10:59:31.571Z")
  },
  "Shipper" : "Fed Ex"
}
```

Zaproponowany powyżej schemat dokumentów określa cechy funkcjonalne elementów znajdujących się w bazie danych. Typy danych zostały tak dobrane, aby umożliwić manipulację danych (np. typ numeryczny lub data gdy jest to potrzebne). W głównej mierze, dane w dokumentach można przedstawić przy pomocy łańcuchów znaków (String). Złożony typ danych wykorzystano przy tworzeniu adresów klienta, a tablicę przy dodawaniu kategorii produktu (tagi). W tabeli orders znajduje się ID Klienta i ID Produktu, dzięki czemu możemy wyszukać który z klientów zamówił jaki produkt. Minusem tego rozwiązania jest fakt, że Klient może zamówić tylko jeden rodzaj produktu w jednym zamówieniu.

Lepiej stworzyć listę produktów w zamówieniu (wraz z Id produktu i ilością).

Stosując poniższe rozwiązanie można dodać kilka produktów do jednego zamówienia.

```
1 //Orders
2
3 {
4   "_id" : ObjectId("5fc2d25733321682d52166d5"),
5   "ClientID" : "5fc2cd2b33321682d52166d3",
6   "Products" : [
7     {
8       "ProductID" : "5fc5353e114ad21a2bc01315",
9       "Units" : NumberInt(1)
10    },
11    {
12      "ProductID" : "123344353e114ad21a2bc01315",
13      "Units" : NumberInt(5)
14    }
15  ],
16  "Dates" : {
17    "Order" : ISODate("2020-09-28T06:01:17.171Z"),
18    "Payment" : ISODate("2020-09-28T11:12:07.271Z"),
19    "Shipped" : ISODate("2020-10-01T08:00:01.271Z"),
20    "Delivery" : ISODate("2020-10-02T10:59:31.571Z")
21  },
22  "Shipper" : "Fed Ex"
23 }
```