

Raport z wykonania ćwiczenia MongoDB

Jakub Płotnikowski

Grudzień 2019r.

Spis treści

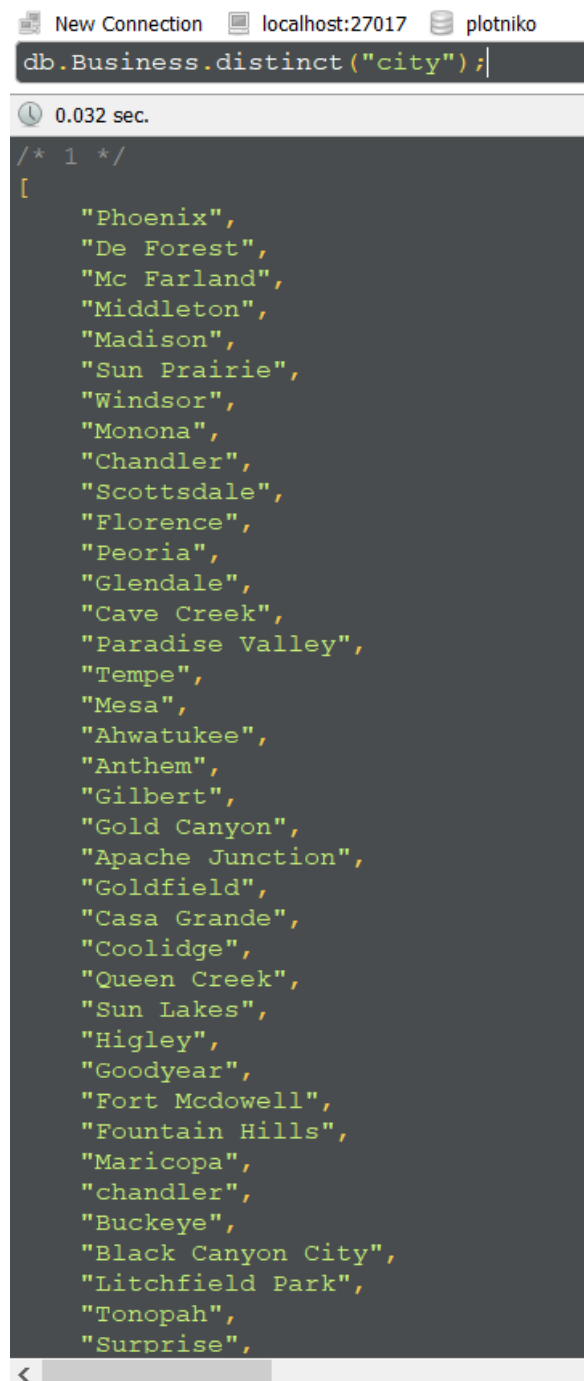
1	Wykorzystując bazę danych yelp dataset wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty:	3
1.a	Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business).	3
1.b	Zwróć liczbę wszystkich recenzji, które pojawiły się w roku 2011 i 2012.	4
1.c	Zwróć dane wszystkich otwartych (open) firm (business) z pól: id, nazwa, adres. .	5
1.d	Zwróć dane wszystkich użytkowników (user), którzy uzyskali przynajmniej jeden pozytywny głos z jednej z kategorii (funny, useful, cool), wynik posortuj alfabetycznie na podstawie imienia użytkownika.	6
1.e	Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2013. Wynik posortuj alfabetycznie na podstawie nazwy firmy.	7
1.f	Wyznacz, jaką średnią ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji, wynik posortuj on najwyższego uzyskanego wyniku.	8
1.g	Usuń wszystkie firmy (business), które posiadają ocenę (stars) poniżej 3.	9
2	Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej wskazówki/napiwku (tip). Wykonaj przykładowe wywołanie.	10
3	Zdefiniuj funkcję (MongoDB), która zwróci wszystkie wskazówki/napiwki (tip), w których w tekście znajdzie się fraza podana jako argument. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.	11
4	Zdefiniuj funkcję (MongoDB), która umożliwi modyfikację nazwy firmy (business) na podstawie id. Id oraz nazwa mają być przekazywane jako parametry.	12
5	Zwróć średnia ilość wszystkich recenzji użytkowników, wykorzystaj map reduce	13
6	Odwzoruj wszystkie zadania z punktu 1 w języku programowania (np. JAVA) z pomocą API do MongoDB. Wykorzystaj dla każdego zadania odrębną metodę.	14
6.a	Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business).	14
6.b	Zwróć liczbę wszystkich recenzji, które pojawiły się w roku 2011 i 2012	14
6.c	Zwróć dane wszystkich otwartych (open) firm (business) z pól: id, nazwa, adres. .	14

6.d	Zwróć dane wszystkich użytkowników (user), którzy uzyskali przynajmniej jeden pozytywny głos z jednej z kategorii (funny, useful, cool), wynik posortuj alfabetycznie na podstawie imienia użytkownika.	15
6.e	Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2013. Wynik posortuj alfabetycznie na podstawie nazwy firmy.	15
6.f	Wyznacz, jaką średnią ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji, wynik posortuj on najwyższego uzyskanego wyniku.	16
6.g	Usuń wszystkie firmy (business), które posiadają ocenę (stars) poniżej 3.	16
6.h	Cały kod klasy MongoDB	17
7	Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: studentów, przedmiotów oraz sal zajęciowych. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych.	20
7.a	Kolekcja Students	20
7.b	Kolekcja Subjects	21
7.c	Kolekcja Classes	22

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

1.a Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business).

```
1 db.Business.distinct("city");
```

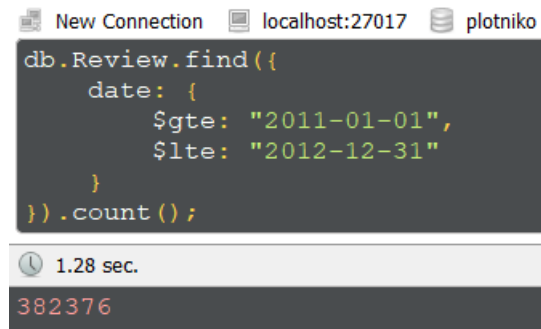


The screenshot shows a MongoDB query interface. At the top, it says "New Connection" and "localhost:27017 plotniko". The query entered is `db.Business.distinct("city");`. Below the query, it shows the execution time "0.032 sec." and the result, which is a list of 40 city names: Phoenix, De Forest, Mc Farland, Middleton, Madison, Sun Prairie, Windsor, Monona, Chandler, Scottsdale, Florence, Peoria, Glendale, Cave Creek, Paradise Valley, Tempe, Mesa, Ahwatukee, Anthem, Gilbert, Gold Canyon, Apache Junction, Goldfield, Casa Grande, Coolidge, Queen Creek, Sun Lakes, Higley, Goodyear, Fort McDowell, Fountain Hills, Maricopa, chandler, Buckeye, Black Canyon City, Litchfield Park, Tonopah, and Surprise.

```
New Connection localhost:27017 plotniko
db.Business.distinct("city");
0.032 sec.
/* 1 */
[
  "Phoenix",
  "De Forest",
  "Mc Farland",
  "Middleton",
  "Madison",
  "Sun Prairie",
  "Windsor",
  "Monona",
  "Chandler",
  "Scottsdale",
  "Florence",
  "Peoria",
  "Glendale",
  "Cave Creek",
  "Paradise Valley",
  "Tempe",
  "Mesa",
  "Ahwatukee",
  "Anthem",
  "Gilbert",
  "Gold Canyon",
  "Apache Junction",
  "Goldfield",
  "Casa Grande",
  "Coolidge",
  "Queen Creek",
  "Sun Lakes",
  "Higley",
  "Goodyear",
  "Fort McDowell",
  "Fountain Hills",
  "Maricopa",
  "chandler",
  "Buckeye",
  "Black Canyon City",
  "Litchfield Park",
  "Tonopah",
  "Surprise",
]
```

1.b Zwróć liczbę wszystkich recenzji, które pojawiły się w roku 2011 i 2012.

```
1 db.Review.find({
2   date: {
3     $gte: "2011-01-01",
4     $lte: "2012-12-31"
5   }
6 }).count();
```



The screenshot shows a MongoDB GUI client window titled 'New Connection' with 'localhost:27017' and 'plotniko' selected. The query editor contains the same MongoDB query as in the previous block. Below the editor, a status bar shows a clock icon and '1.28 sec.'. At the bottom, the result is displayed as the number '382376'.

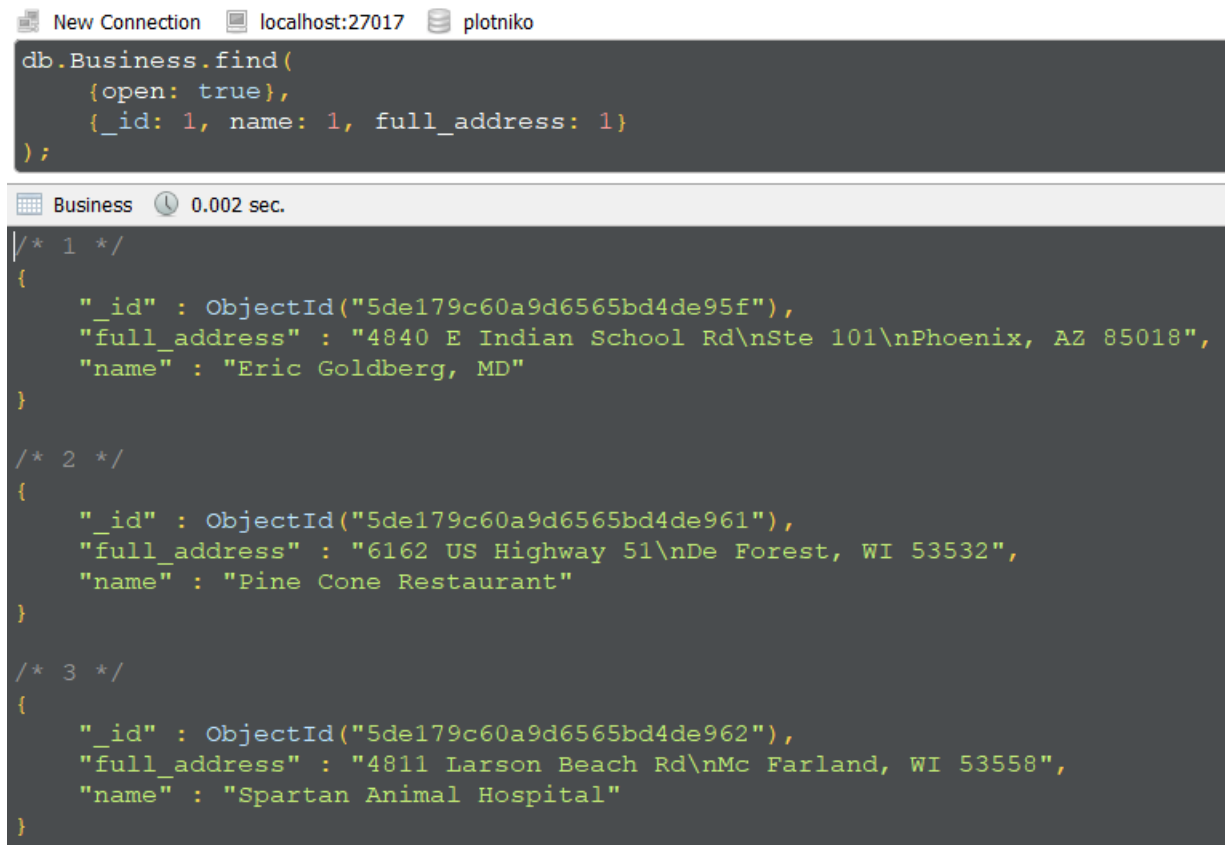
```
db.Review.find({
  date: {
    $gte: "2011-01-01",
    $lte: "2012-12-31"
  }
}).count();
```

1.28 sec.

382376

1.c Zwróć dane wszystkich otwartych (open) firm (business) z pól: id, nazwa, adres.

```
1 db.Business.find(  
2   {open: true},  
3   {_id: 1, name: 1, full_address: 1}  
4 );
```



The screenshot shows a MongoDB web interface. At the top, it says 'New Connection' with 'localhost:27017' and 'plotniko'. Below that is a code editor with the same query as in the previous block. The results are displayed in a table with the title 'Business' and a execution time of '0.002 sec.'. The results are shown in a JSON format, with three documents. Each document has fields for '_id', 'full_address', and 'name'.

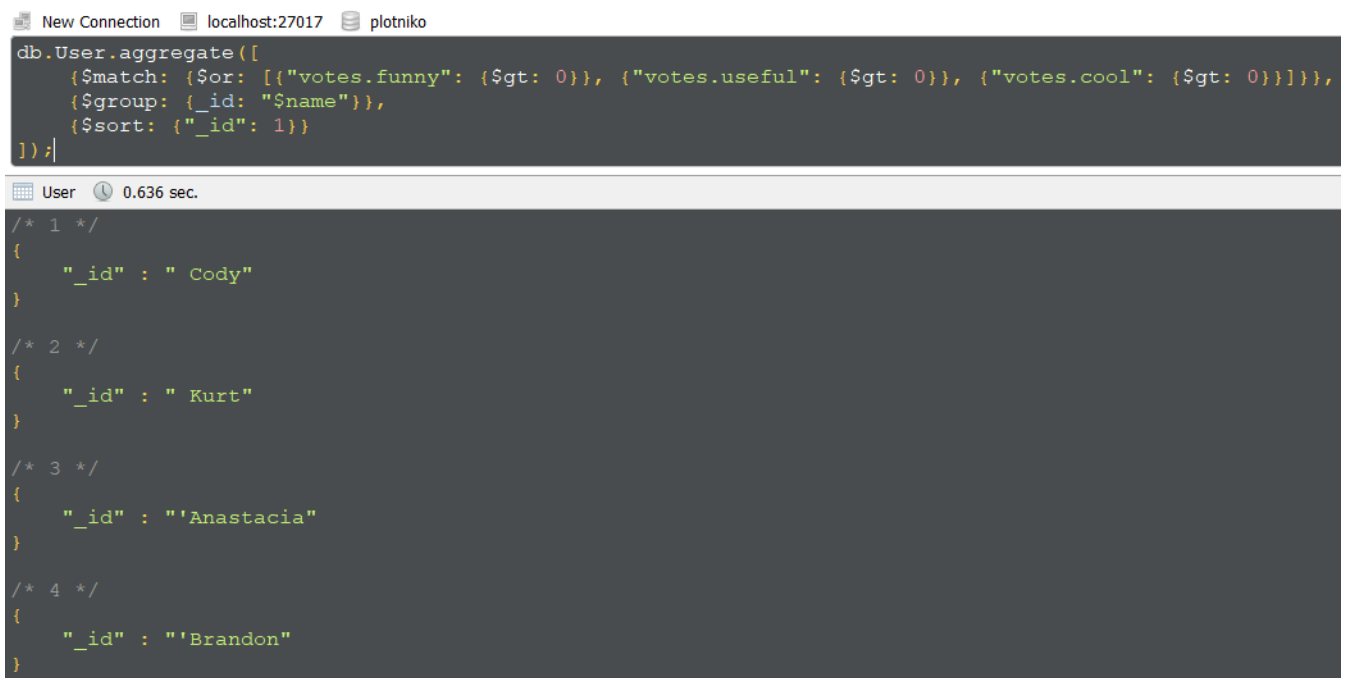
```
db.Business.find(  
  {open: true},  
  {_id: 1, name: 1, full_address: 1}  
);
```

Business 0.002 sec.

```
/* 1 */  
{  
  "_id" : ObjectId("5de179c60a9d6565bd4de95f"),  
  "full_address" : "4840 E Indian School Rd\nSte 101\nPhoenix, AZ 85018",  
  "name" : "Eric Goldberg, MD"  
}  
  
/* 2 */  
{  
  "_id" : ObjectId("5de179c60a9d6565bd4de961"),  
  "full_address" : "6162 US Highway 51\nDe Forest, WI 53532",  
  "name" : "Pine Cone Restaurant"  
}  
  
/* 3 */  
{  
  "_id" : ObjectId("5de179c60a9d6565bd4de962"),  
  "full_address" : "4811 Larson Beach Rd\nMc Farland, WI 53558",  
  "name" : "Spartan Animal Hospital"  
}
```

- 1.d Zwróć dane wszystkich użytkowników (user), którzy uzyskali przynajmniej jeden pozytywny głos z jednej z kategorii (funny, useful, cool), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

```
1 db.User.aggregate([
2   {$match: {$or: [{"votes.funny": {$gt: 0}}, {"votes.useful": {
3     $gt: 0}}, {"votes.cool": {$gt: 0}}]}},
4   {$group: {_id: "$name"}},
5   {$sort: {"_id": 1}}
6 ]);
```



The screenshot shows a MongoDB query client interface. At the top, there's a header with 'New Connection', 'localhost:27017', and a user icon labeled 'plotniko'. Below this, the query is entered in a text area:

```
db.User.aggregate([
  {$match: {$or: [{"votes.funny": {$gt: 0}}, {"votes.useful": {$gt: 0}}, {"votes.cool": {$gt: 0}}]}},
  {$group: {_id: "$name"}},
  {$sort: {"_id": 1}}
]);
```

Below the query area, a status bar shows 'User' and '0.636 sec.'. The results are displayed in a list format with four entries, each preceded by a comment indicating its index:

```
/* 1 */
{
  "_id" : "Cody"
}

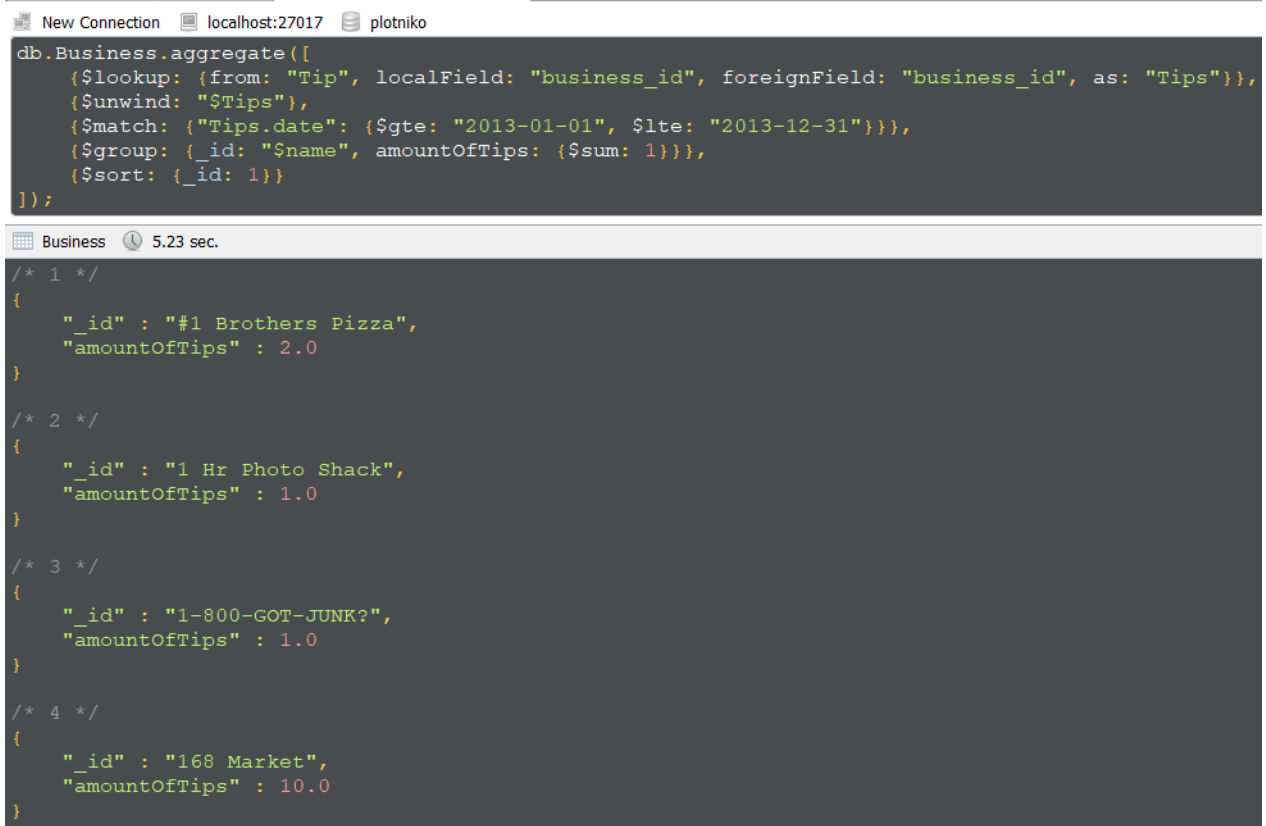
/* 2 */
{
  "_id" : "Kurt"
}

/* 3 */
{
  "_id" : "'Anastacia"
}

/* 4 */
{
  "_id" : "'Brandon"
}
```

- 1.e Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2013. Wynik posortuj alfabetycznie na podstawie nazwy firmy.

```
1 db.Business.aggregate([
2   {$lookup: {from: "Tip", localField: "business_id",
3     foreignField: "business_id", as: "Tips"}},
4   {$unwind: "$Tips"},
5   {$match: {"Tips.date": {$gte: "2013-01-01", $lte: "2013-12-31"
6     }}}},
7   {$group: {_id: "$name", amountOfTips: {$sum: 1}}},
8   {$sort: {_id: 1}}
9 ]);
```



```
db.Business.aggregate([
  {$lookup: {from: "Tip", localField: "business_id", foreignField: "business_id", as: "Tips"}},
  {$unwind: "$Tips"},
  {$match: {"Tips.date": {$gte: "2013-01-01", $lte: "2013-12-31"}}},
  {$group: {_id: "$name", amountOfTips: {$sum: 1}}},
  {$sort: {_id: 1}}
]);
```

Business 5.23 sec.

```
/* 1 */
{
  "_id" : "#1 Brothers Pizza",
  "amountOfTips" : 2.0
}

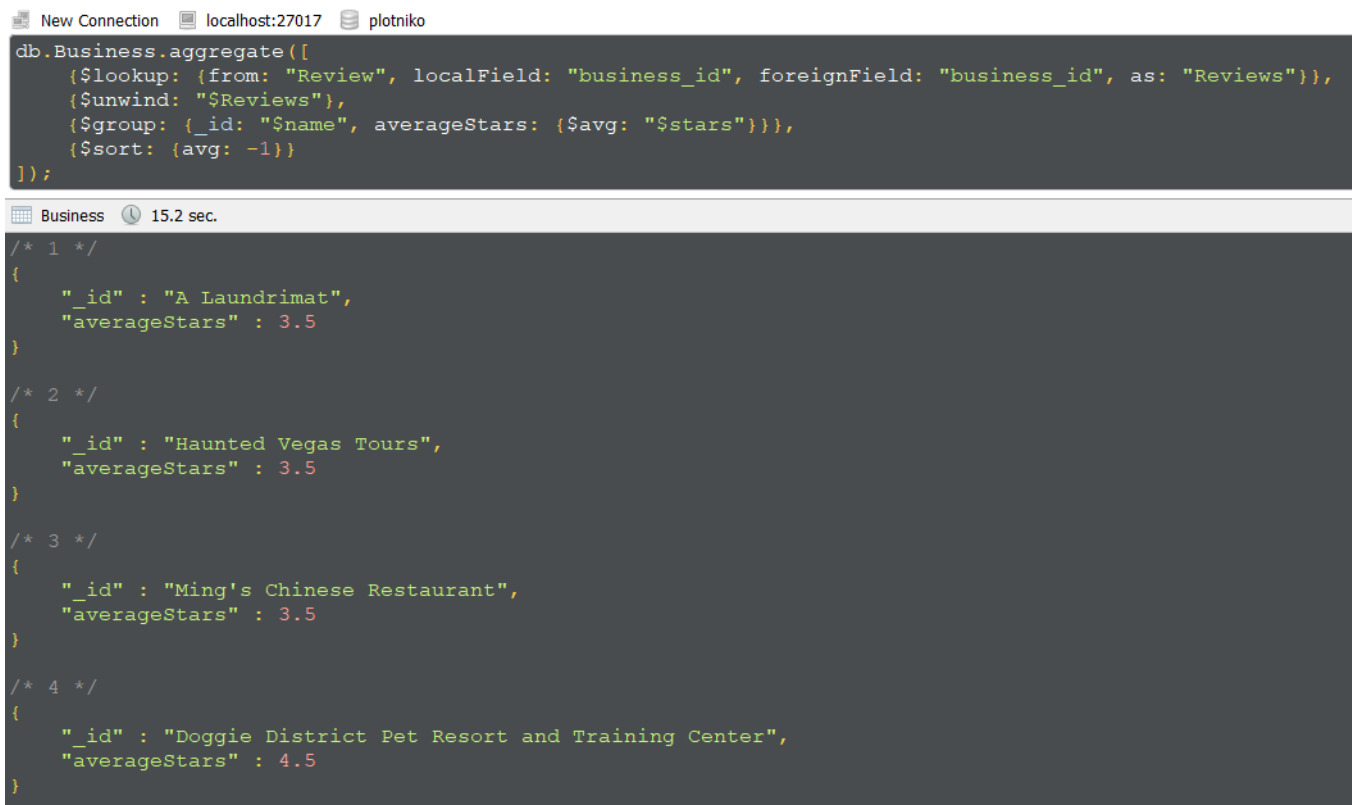
/* 2 */
{
  "_id" : "1 Hr Photo Shack",
  "amountOfTips" : 1.0
}

/* 3 */
{
  "_id" : "1-800-GOT-JUNK?",
  "amountOfTips" : 1.0
}

/* 4 */
{
  "_id" : "168 Market",
  "amountOfTips" : 10.0
}
```

- 1.f Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji, wynik posortuj on najwyższego uzyskanego wyniku.

```
1 db.Business.aggregate([
2   {$lookup: {from: "Review", localField: "business_id",
3     foreignField: "business_id", as: "Reviews"}}},
4   {$unwind: "$Reviews"},
5   {$group: {_id: "$name", averageStars: {$avg: "$stars"}}},
6   {$sort: {avg: -1}}
7 ]);
```



The screenshot shows a MongoDB query interface. At the top, there's a header with "New Connection", "localhost:27017", and "plotniko". Below this, the query is entered into a text area:

```
db.Business.aggregate([
  {$lookup: {from: "Review", localField: "business_id", foreignField: "business_id", as: "Reviews"}},
  {$unwind: "$Reviews"},
  {$group: {_id: "$name", averageStars: {$avg: "$stars"}}},
  {$sort: {avg: -1}}
]);
```

Below the query area, there's a status bar showing "Business" and a clock icon with "15.2 sec.". The results are displayed in a JSON format, showing the top 4 businesses by average stars:

```
/* 1 */
{
  "_id" : "A Laundrimat",
  "averageStars" : 3.5
}

/* 2 */
{
  "_id" : "Haunted Vegas Tours",
  "averageStars" : 3.5
}

/* 3 */
{
  "_id" : "Ming's Chinese Restaurant",
  "averageStars" : 3.5
}

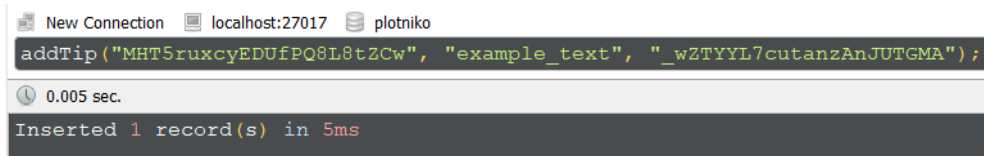
/* 4 */
{
  "_id" : "Doggie District Pet Resort and Training Center",
  "averageStars" : 4.5
}
```


1.g Usuń wszystkie firmy (business), które posiadają ocenę (stars) poniżej 3.

```
1 var businesses = db.Business.aggregate([
2   {$lookup: {from: "Review", localField: "business_id",
3     foreignField: "business_id", as: "Reviews"}},
4   {$unwind: "$Reviews"},
5   {$group: {_id: "$name", averageStars: {$avg: "$stars"}}},
6   {$sort: {averageStars: -1}}
7 ]);
8 businesses.forEach(function (business) {
9   if (business.avg < 3) {
10     db.Business.remove({"name": business._id});
11   }
12 });
```

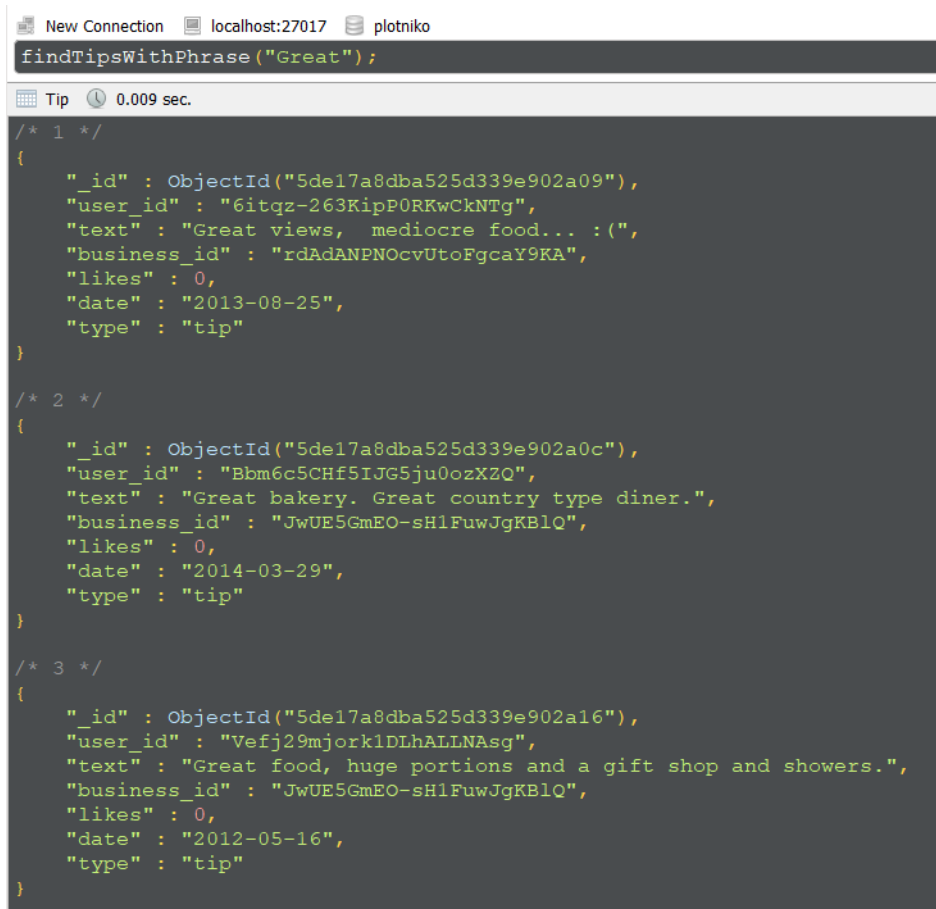
2 Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej wskazówki/napiwku (tip). Wykonaj przykładowe wywołanie.

```
1 function addTip(user_id, text, business_id) {
2   db.Tip.insert({
3     user_id: user_id,
4     text: text,
5     business_id: business_id,
6     likes: 0,
7     date: ISODate(),
8     type: "tip"
9   });
10 }
11
12 addTip("MHT5ruxcyEDUfPQ8L8tZCw", "example_text", "
    _wZTYYL7cutanzAnJUTGMA");
```



- 3 Zdefiniuj funkcję (MongoDB), która zwróci wszystkie wskazówki/napiwki (tip), w których w tekście znajdzie się fraza podana jako argument. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.

```
1 function findTipsWithPhrase(phrase) {
2     return db.Tip.find({text: new RegExp(phrase)})
3 }
4
5 findTipsWithPhrase("Great");
```



The screenshot shows a web interface for a MongoDB connection. At the top, it says 'New Connection', 'localhost:27017', and 'plotniko'. Below that, the command 'findTipsWithPhrase("Great");' is entered. The result is displayed in a table with one column 'Tip' and a value of '0.009 sec.'. The table contains three rows of JSON data, each representing a tip. The first row is a tip about 'Great views, mediocre food... :(' from user '6itqz-263KipP0RKwCkNTg' at business 'rdAdANPN0cvUtoFgcaY9KA' on 2013-08-25. The second row is a tip about 'Great bakery. Great country type diner.' from user 'Bbm6c5CHf5IJG5ju0ozXZQ' at business 'JwUE5GmEO-sH1FuwJgKB1Q' on 2014-03-29. The third row is a tip about 'Great food, huge portions and a gift shop and showers.' from user 'Vefj29mjork1DLhALLNAsg' at business 'JwUE5GmEO-sH1FuwJgKB1Q' on 2012-05-16.

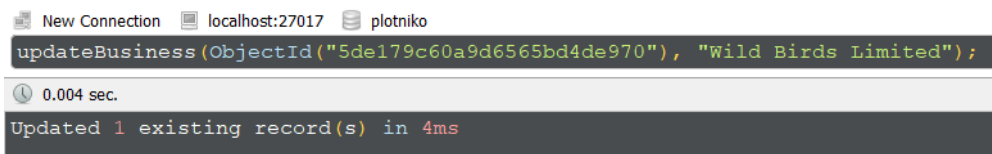
```
/* 1 */
{
  "_id" : ObjectId("5de17a8dba525d339e902a09"),
  "user_id" : "6itqz-263KipP0RKwCkNTg",
  "text" : "Great views, mediocre food... :(",
  "business_id" : "rdAdANPN0cvUtoFgcaY9KA",
  "likes" : 0,
  "date" : "2013-08-25",
  "type" : "tip"
}

/* 2 */
{
  "_id" : ObjectId("5de17a8dba525d339e902a0c"),
  "user_id" : "Bbm6c5CHf5IJG5ju0ozXZQ",
  "text" : "Great bakery. Great country type diner.",
  "business_id" : "JwUE5GmEO-sH1FuwJgKB1Q",
  "likes" : 0,
  "date" : "2014-03-29",
  "type" : "tip"
}

/* 3 */
{
  "_id" : ObjectId("5de17a8dba525d339e902a16"),
  "user_id" : "Vefj29mjork1DLhALLNAsg",
  "text" : "Great food, huge portions and a gift shop and showers.",
  "business_id" : "JwUE5GmEO-sH1FuwJgKB1Q",
  "likes" : 0,
  "date" : "2012-05-16",
  "type" : "tip"
}
```

- 4 Zdefiniuj funkcję (MongoDB), która umożliwi modyfikację nazwy firmy (business) na podstawie id. Id oraz nazwa mają być przekazywane jako parametry.

```
1 function updateBusiness(id, new_business_name) {  
2     db.Business.update(  
3         {_id: id},  
4         {$set: {name: new_business_name}}  
5     )  
6 }  
7  
8 updateBusiness(ObjectId("5de179c60a9d6565bd4de970"), "Wild Birds  
    Limited");
```



The screenshot shows a MongoDB command prompt interface. At the top, it displays 'New Connection', 'localhost:27017', and 'plotniko'. The command entered is `updateBusiness(ObjectId("5de179c60a9d6565bd4de970"), "Wild Birds Limited");`. Below the command, a status bar indicates the execution time as '0.004 sec.' and the result as 'Updated 1 existing record(s) in 4ms'.

5 Zwróć średnia ilość wszystkich recenzji użytkowników, wykorzystaj map reduce

```
1 var map = function () {
2     emit(this.user_id, 1);
3 };
4
5 var reduce = function (user_id, reviews) {
6     return Array.sum(reviews);
7 };
8
9 db.Review.mapReduce(
10     map,
11     reduce,
12     {out: "Reviews"}
13 );
14
15 db.Reviews.aggregate([
16     {$group: {_id: null, avg: {$avg: "$value"}}}
17 ]);
```

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

6.a Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business).

```
1 private void executeTask1a() {
2     DistinctIterable uniqueCities = businessCollection.distinct("
3         city", String.class);
4
5     for (Object city : uniqueCities) {
6         System.out.println(city);
7     }
8 }
```

6.b Zwróć liczbę wszystkich recenzji, które pojawiły się w roku 2011 i 2012

```
1 private void executeTask1b() {
2     Bson dateRange = and(
3         gte("date", "2011-01-01"),
4         lte("date", "2012-12-31"));
5
6     long count = reviewCollection.countDocuments(dateRange);
7
8     System.out.println(count);
9 }
```

6.c Zwróć dane wszystkich otwartych (open) firm (business) z pól: id, nazwa, adres.

```
1 private void executeTask1c() {
2     FindIterable result = businessCollection
3         .find(eq("open", true))
4         .projection(include("_id", "name", "full_address"));
5
6     for (Object o : result) {
7         System.out.println(o);
8     }
9 }
```

- 6.d Zwróć dane wszystkich użytkowników (user), którzy uzyskali przynajmniej jeden pozytywny głos z jednej z kategorii (funny, useful, cool), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

```
1 private void executeTask1d() {
2     Bson filter = or(
3         gt("votes.funny", 0),
4         gt("votes.useful", 0),
5         gt("votes.cool", 0)
6     );
7
8     AggregateIterable result = userCollection.aggregate(
9         asList(
10             match(filter),
11             group("$name"),
12             sort(ascending("_id"))
13         ));
14
15     for (Object o : result) {
16         System.out.println(o);
17     }
18 }
```

- 6.e Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2013. Wynik posortuj alfabetycznie na podstawie nazwy firmy.

```
1 private void executeTask1e() {
2     AggregateIterable result = businessCollection.aggregate(
3         asList(
4             lookup("Tip", "business_id", "business_id", "
5                 Tips"),
6             unwind("$Tips"),
7             match(and(
8                 gte("Tips.date", "2013-01-01"),
9                 lte("Tips.date", "2013-12-31")
10             )),
11             group("$name", sum("amountOfTips", 1)),
12             sort(ascending("_id"))
13         ));
14
15     for (Object o : result) {
16         System.out.println(o);
17     }
18 }
```

6.f Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji, wynik posortuj on najwyższego uzyskanego wyniku.

```
1 private void executeTask1f() {
2     AggregateIterable result = businessCollection.aggregate(
3         asList(
4             lookup("Review", "business_id", "business_id",
5                 "Reviews"),
6             unwind("$Reviews"),
7             group("$name", avg("averageStars", "$stars")),
8             sort(descending("averageStars"))
9         ));
10    for (Object o : result) {
11        System.out.println(o);
12    }
13 }
```

6.g Usuń wszystkie firmy (business), które posiadają ocenę (stars) poniżej 3.

```
1 private void executeTask1g() {
2     AggregateIterable<Document> aggregateIterable =
3         businessCollection.aggregate(
4             asList(
5                 lookup("Review", "business_id", "business_id",
6                     "Reviews"),
7                 unwind("$Reviews"),
8                 group("$name", avg("averageStars", "$stars")),
9                 match(lt("averageStars", 3))
10            ));
11    for (Document document : aggregateIterable) {
12        Document documentToBeDeleted = new Document(new Document("
13            name", document.getString("_id")));
14        DeleteResult deleteResult = businessCollection.deleteMany(
15            documentToBeDeleted);
16        System.out.println(deleteResult);
17    }
18 }
```


6.h Cały kod klasy MongoDB

```
1 public class MongoDB {
2
3     private MongoCollection<Document> businessCollection;
4     private MongoCollection<Document> reviewCollection;
5     private MongoCollection<Document> userCollection;
6
7     private MongoDB() {
8         MongoDBDatabase database = new MongoClient().getDatabase("
9             plotniko");
10
11         businessCollection = database.getCollection("Business");
12         reviewCollection = database.getCollection("Review");
13         userCollection = database.getCollection("User");
14     }
15
16     public static void main(String[] args) {
17         MongoDB mongoDB = new MongoDB();
18         mongoDB.executeTask1a();
19         mongoDB.executeTask1b();
20         mongoDB.executeTask1c();
21         mongoDB.executeTask1d();
22         mongoDB.executeTask1e();
23         mongoDB.executeTask1f();
24         mongoDB.executeTask1g();
25     }
26
27     private void executeTask1a() {
28         DistinctIterable<String> uniqueCities = businessCollection
29             .distinct("city", String.class);
30
31         for (Object city : uniqueCities) {
32             System.out.println(city);
33         }
34     }
35
36     private void executeTask1b() {
37         Bson dateRange = and(
38             gte("date", "2011-01-01"),
39             lte("date", "2012-12-31"));
40
41         long count = reviewCollection.countDocuments(dateRange);
42
43         System.out.println(count);
44     }
45
46     private void executeTask1c() {
47         FindIterable<Document> result = businessCollection
48             .find(eq("open", true))
```

```

47         .projection(include("_id", "name", "full_address")
48         );
49     for (Object o : result) {
50         System.out.println(o);
51     }
52 }
53
54 private void executeTask1d() {
55     Bson filter = or(
56         gt("votes.funny", 0),
57         gt("votes.useful", 0),
58         gt("votes.cool", 0)
59     );
60
61     AggregateIterable<Document> result = userCollection.
62         aggregate(
63             asList(
64                 match(filter),
65                 group("$name"),
66                 sort(ascending("_id"))
67             ));
68
69     for (Object o : result) {
70         System.out.println(o);
71     }
72
73 private void executeTask1e() {
74     AggregateIterable<Document> result = businessCollection.
75         aggregate(
76             asList(
77                 lookup("Tip", "business_id", "business_id",
78                     , "Tips"),
79                 unwind("$Tips"),
80                 match(and(
81                     gte("Tips.date", "2013-01-01"),
82                     lte("Tips.date", "2013-12-31")
83                 )),
84                 group("$name", sum("amountOfTips", 1)),
85                 sort(ascending("_id"))
86             ));
87
88     for (Object o : result) {
89         System.out.println(o);
90     }
91
92 private void executeTask1f() {
93     AggregateIterable<Document> result = businessCollection.
94         aggregate(

```

```

93         asList(
94             lookup("Review", "business_id", "
                business_id", "Reviews"),
95             unwind("$Reviews"),
96             group("$name", avg("averageStars", "$stars
                "))),
97             sort(descending("averageStars")))
98         ));
99
100     for (Object o : result) {
101         System.out.println(o);
102     }
103 }
104
105 private void executeTask1g() {
106     AggregateIterable<Document> aggregateIterable =
107         businessCollection.aggregate(
108             asList(
109                 lookup("Review", "business_id", "
                    business_id", "Reviews"),
110                 unwind("$Reviews"),
111                 group("$name", avg("averageStars", "$stars
                    "))),
112                 match(lt("averageStars", 3))
113             );
114
115     for (Document document : aggregateIterable) {
116         Document documentToBeDeleted = new Document(new
            Document("name", document.getString("_id")));
117         DeleteResult deleteResult = businessCollection.
            deleteMany(documentToBeDeleted);
118         System.out.println(deleteResult);
119     }
120 }
121 }

```

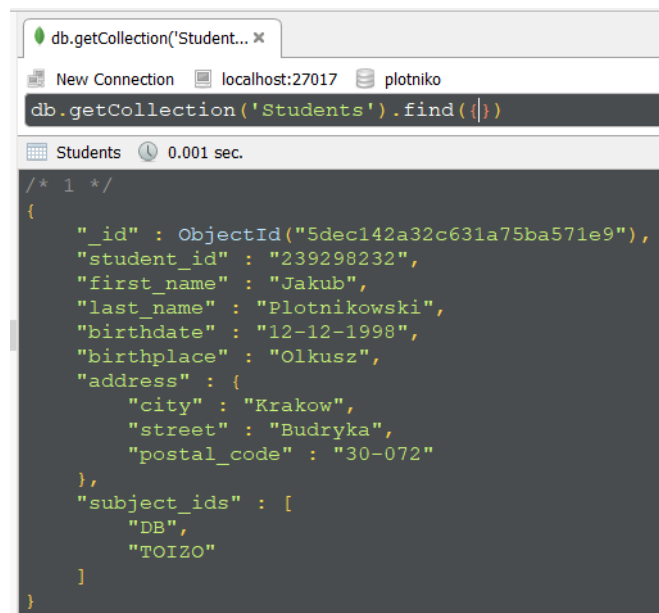
- 7 Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: studentów, przedmiotów oraz sal zajęciowych. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych.

7.a Kolekcja Students

```
1 db.createCollection("Students");

1 db.Students.insert({
2   "student_id": "239298232",
3   "first_name": "Jakub",
4   "last_name": "Plotnikowski",
5   "birthdate": "12-12-1998",
6   "birthplace": "Olkusz",
7   "address": {
8     "city": "Krakow",
9     "street": "Budryka",
10    "postal_code": "30-072"
11  },
12  "subject_ids": ["DB", "TOIZO"]
13 });
```

Przykład pobrania danych z kolekcji Students:



The screenshot shows a MongoDB command prompt window. The title bar indicates the connection is to 'localhost:27017' with the database 'plotniko'. The command entered is `db.getCollection('Students').find({})`. The execution time is 0.001 sec. The result is a JSON document representing a student:

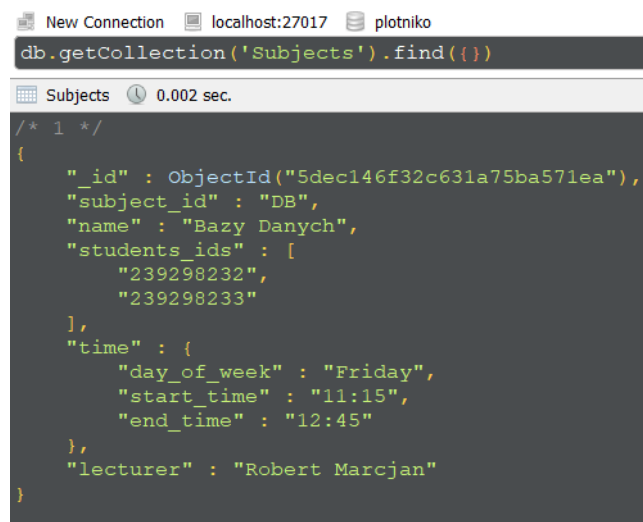
```
/* 1 */
{
  "_id" : ObjectId("5dec142a32c631a75ba571e9"),
  "student_id" : "239298232",
  "first_name" : "Jakub",
  "last_name" : "Plotnikowski",
  "birthdate" : "12-12-1998",
  "birthplace" : "Olkusz",
  "address" : {
    "city" : "Krakow",
    "street" : "Budryka",
    "postal_code" : "30-072"
  },
  "subject_ids" : [
    "DB",
    "TOIZO"
  ]
}
```

7.b Kolekcja Subjects

```
1 db.createCollection("Subjects");

1 db.Subjects.insert({
2   "subject_id": "DB",
3   "name": "Bazy Danych",
4   "students_ids": ["239298232", "239298233"],
5   "time": {
6     "day_of_week": "Friday",
7     "start_time": "11:15",
8     "end_time": "12:45"
9   },
10  "lecturer": "Robert Marcjan"
11 });
```

Przykład pobrania danych z kolekcji Subjects:



The screenshot shows a MongoDB command prompt window. At the top, it says "New Connection", "localhost:27017", and "plotniko". The command entered is `db.getCollection('Subjects').find({})`. Below the command, it says "Subjects" and "0.002 sec.". The output is a JSON document:

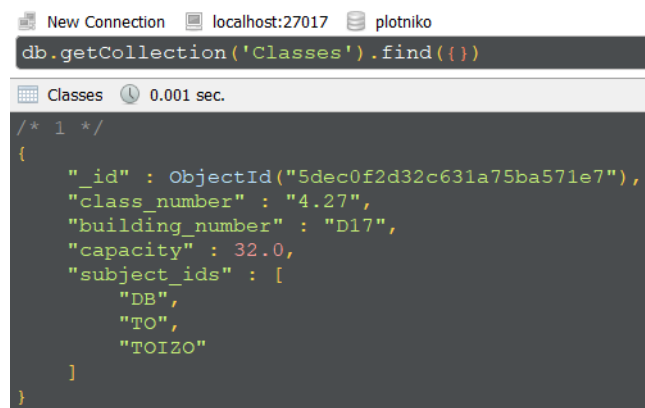
```
/* 1 */
{
  "_id" : ObjectId("5dec146f32c631a75ba571ea"),
  "subject_id" : "DB",
  "name" : "Bazy Danych",
  "students_ids" : [
    "239298232",
    "239298233"
  ],
  "time" : {
    "day_of_week" : "Friday",
    "start_time" : "11:15",
    "end_time" : "12:45"
  },
  "lecturer" : "Robert Marcjan"
}
```

7.c Kolekcja Classes

```
1 db.createCollection("Classes");

1 db.Classes.insert({
2     "class_number": "4.27",
3     "building_number": "D17",
4     "capacity": 32,
5     "subject_ids": ["DB", "TO", "TOIZO"]
6 });
```

Przykład pobrania danych z kolekcji Classes:



The screenshot shows a MongoDB query result in a GUI tool. At the top, it says 'New Connection', 'localhost:27017', and 'plotniko'. Below that, the query 'db.getCollection('Classes').find({})' is entered. The result is displayed in a table with the title 'Classes' and a duration of '0.001 sec.'. The table contains one document with the following fields: '_id' (ObjectId), 'class_number' (4.27), 'building_number' (D17), 'capacity' (32.0), and 'subject_ids' (an array containing 'DB', 'TO', and 'TOIZO').

```
/* 1 */
{
  "_id" : ObjectId("5dec0f2d32c631a75ba571e7"),
  "class_number" : "4.27",
  "building_number" : "D17",
  "capacity" : 32.0,
  "subject_ids" : [
    "DB",
    "TO",
    "TOIZO"
  ]
}
```