

Raport z laboratorium z przedmiotu Bazy Danych MongoDB

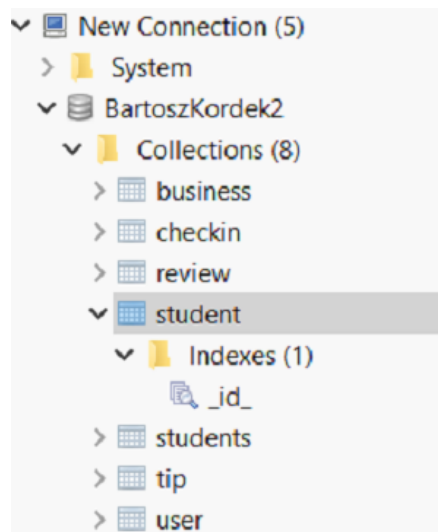
Bartosz Kordek
Informatyka, studia niestacjonarne, semestr VII

4) Za pomocą narzędzia Robo 3T wykonaj polecenie dodające do stworzonej bazy kolekcję „student”:

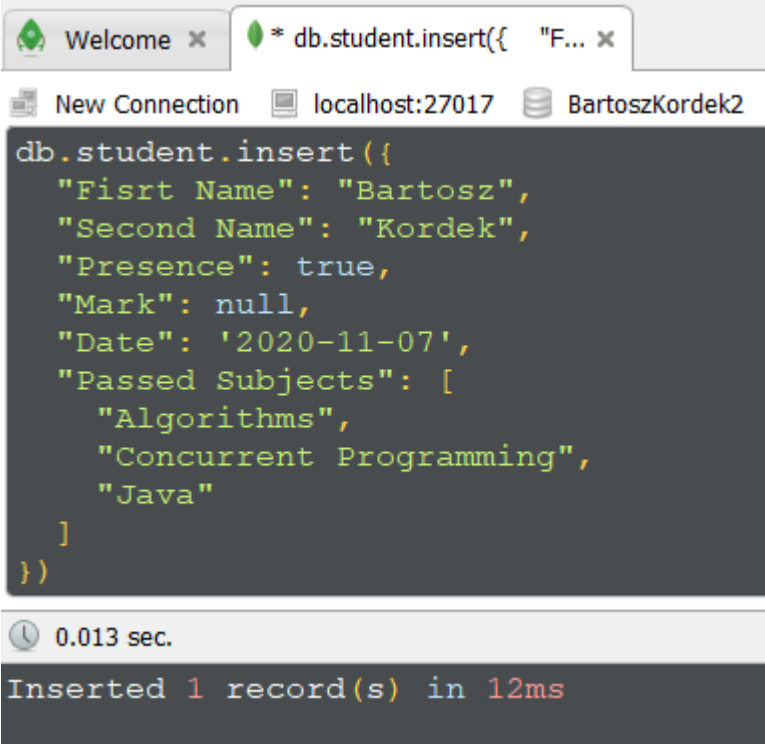


Wynik:

Została utworzona nowa kolekcja “student”



- a) wprowadź własne dane do kolekcji: imię, nazwisko, obecność (typ bool), ocena z lab. (null), aktualna data, zaliczone przedmioty (min 3 przykładowe).



The screenshot shows a MongoDB command prompt window with the following content:

```
Welcome x * db.student.insert({ "F... x
```

Below the command prompt, there is a status bar showing "New Connection", "localhost:27017", and "BartoszKordek2".

```
db.sturt.insert({
  "Fisrt Name": "Bartosz",
  "Second Name": "Kordek",
  "Presence": true,
  "Mark": null,
  "Date": '2020-11-07',
  "Passed Subjects": [
    "Algorithms",
    "Concurrent Programming",
    "Java"
  ]
})
```

Below the command prompt, there is a status bar showing "0.013 sec." and "Inserted 1 record(s) in 12ms".

Na podstawie komunikatu widać, że rekord został dodany do bazy danych.

To samo zadanie można wykonać za pomocą funkcji napisanej np. w języku Java.

```
private void insertStudent() {
    DBCollection gettedCollection = db.getCollection("student");
    Document student = new Document("_id", new ObjectId());
    student.append("First Name", "John")
        .append("Last Name", "Smith")
        .append("Presence", new Boolean(true))
        .append("Mark", null)
        .append("Current Date", "2020-11-07")
        .append("Passed Subjects", asList("JAVA", "CONCURRENT PROGRAMMING", "ALGORITHMS"));

    BasicDBObject basicDBObject = new BasicDBObject(student);
    gettedCollection.insert(basicDBObject);
}
```

Wynik - wstawiony dokument ze studentem Johnem Smith.

```
db.getCollection('student').find({})
```

View Document

localhost:27017 BartoszKordek2 student

```
{
  "_id" : ObjectId("5fa701c68c5b347790c643bf"),
  "First Name" : "John",
  "Last Name" : "Smith",
  "Presence" : true,
  "Mark" : null,
  "Current Date" : "2020-11-07",
  "Passed Subjects" : [
    "JAVA",
    "CONCURRENT PROGRAMMING",
    "ALGORITHMS"
  ]
}
```

Zauważyłem, że data wpisana w Javie nie parsuje się dobrze do dokumentu, jeżeli została stworzona przy pomocy obiektu Date, np. ("Current Date", new Date(2020-11-06)) lub ("Current Date", new Date(2020, 11, 07)).

Uzyskane wyniki:

- ```
.append("Current Date", new Date(2020-11-06))
```

```
Mark : null,
"Current Date" : ISODate("1970-01-01T00:00:02.003Z"),
"Passed Subjects" : [
```
- ```
.append("Current Date", new Date(2020,11,07))
```

```
Mark : null,
"Current Date" : Date(615654396000000),
```

b) wyświetl wynik dodania danej w formie. json txt

W celu wyświetlenia wszystkich dokumentów znajdujących się w kolekcji "student" należy wpisać poniższą komendę w terminalu.

```
New Connection localhost:27017 BartoszKordek2
```

```
db.getCollection('student').find({})
```


student 0.002 sec.

Można wyszukać wstawiony dokument:

```
db.getCollection('student').find({})
```

Key	Value	Type
(1) ObjectId("5fa67c83167c1a12e7e07a7c")	{ 7 fields }	Object
_id	ObjectId("5fa67c83167c1a12e7e07a7c")	ObjectId
First Name	Bartosz	String
Last Name	Kordek	String
Presence	true	Boolean
Mark	null	Null
Current Date	2020-11-06	String
Passed Subjects	[3 elements]	Array
[0]	JAVA	String
[1]	CONCURRENT PROGRAMMING	String
[2]	ALGORITHMS	String

Oraz wyświetlić wstawione dane w formie dokumentu w formacie json:

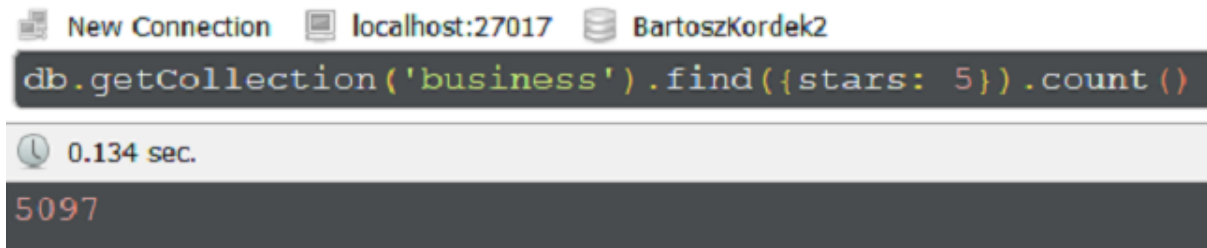
 View Document

```
localhost:27017  BartoszKordek2  student

{
  "_id" : ObjectId("5fa67c83167c1a12e7e07a7c"),
  "First Name" : "Bartosz",
  "Last Name" : "Kordek",
  "Presence" : true,
  "Mark" : null,
  "Current Date" : "2020-11-06",
  "Passed Subjects" : [
    "JAVA",
    "CONCURRENT PROGRAMMING",
    "ALGORITHMS"
  ]
}
```

5) Za pomocą narzędzia Robo 3T wykonaj zapytania, które pozwolą uzyskać następujące wyniki:

- a) ilość miejsc ocenianych na 5 gwiazdek (*pole stars, kolekcja business*)



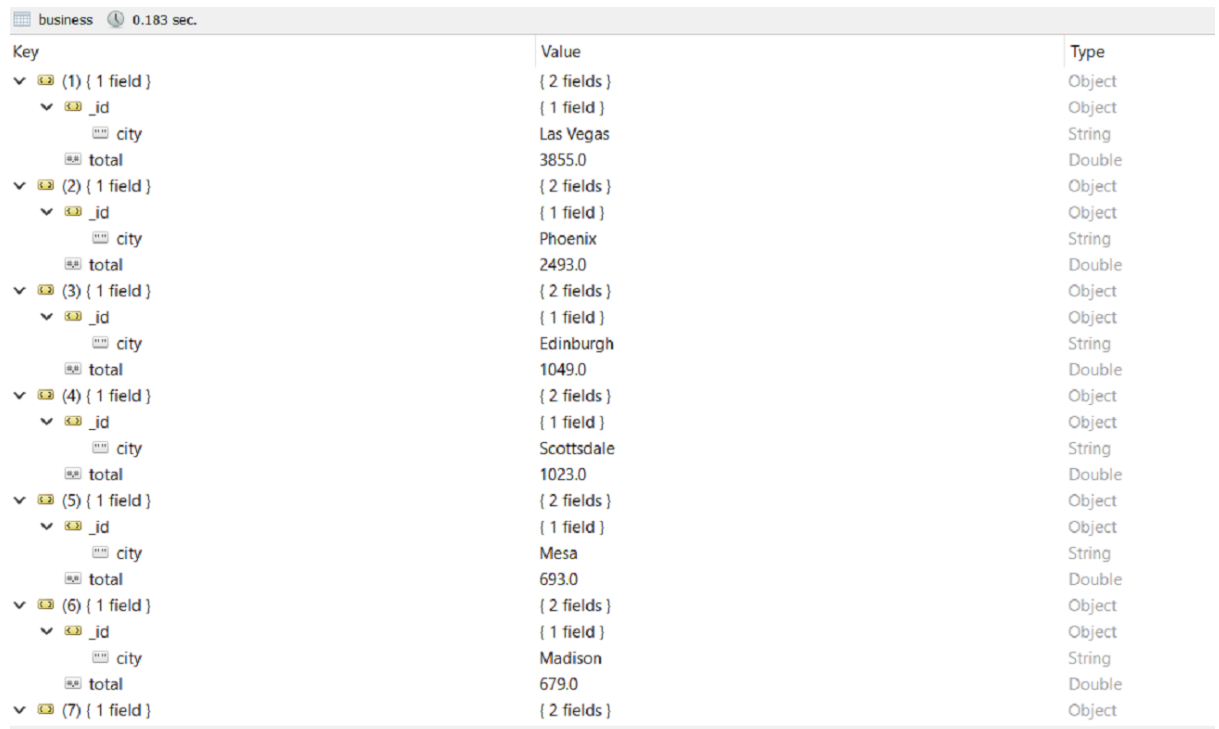
The screenshot shows the Robo 3T interface with a new connection to localhost:27017. The query entered is `db.getCollection('business').find({stars: 5}).count()`. The execution time is 0.134 sec. and the result is 5097.

- b) ilość restauracji w każdym mieście, wynik posortuj malejąco na podstawie liczby. Pole categories w dokumencie business musi zawierać wartość Restaurants.



The screenshot shows the Robo 3T interface with a new connection to localhost:27017. The query entered is `db.getCollection('business').aggregate([{ $match: { "categories": "Restaurants" } }, { $group: { _id: { city: "$city" }, total: { $sum: 1 } } }, { $sort: { total: -1 } }])`. The execution time is 0.183 sec.

Wyniki:



Key	Value	Type
✓ (1) { 1 field }	{ 2 fields }	Object
└─ _id	{ 1 field }	Object
└─ city	Las Vegas	String
└─ total	3855.0	Double
✓ (2) { 2 fields }	{ 2 fields }	Object
└─ _id	{ 1 field }	Object
└─ city	Phoenix	String
└─ total	2493.0	Double
✓ (3) { 3 fields }	{ 2 fields }	Object
└─ _id	{ 1 field }	Object
└─ city	Edinburgh	String
└─ total	1049.0	Double
✓ (4) { 4 fields }	{ 2 fields }	Object
└─ _id	{ 1 field }	Object
└─ city	Scottsdale	String
└─ total	1023.0	Double
✓ (5) { 5 fields }	{ 2 fields }	Object
└─ _id	{ 1 field }	Object
└─ city	Mesa	String
└─ total	693.0	Double
✓ (6) { 6 fields }	{ 2 fields }	Object
└─ _id	{ 1 field }	Object
└─ city	Madison	String
└─ total	679.0	Double
✓ (7) { 7 fields }	{ 2 fields }	Object

Zauważyłem, że po użyciu funkcji `aggregate`, nie działa funkcja `count`.

- c) ilość hoteli (atrybut categories powinien mieć wartość Hotels) w każdym stanie/okręgu (state), które posiadają darmowe Wi-fi (pole attributes, klucz-wartość 'Wi-Fi': 'free') oraz ocenę co najmniej 4.5 gwiazdki. Wykorzystaj funkcję group.

New Connection localhost:27017 BartoszKordek2

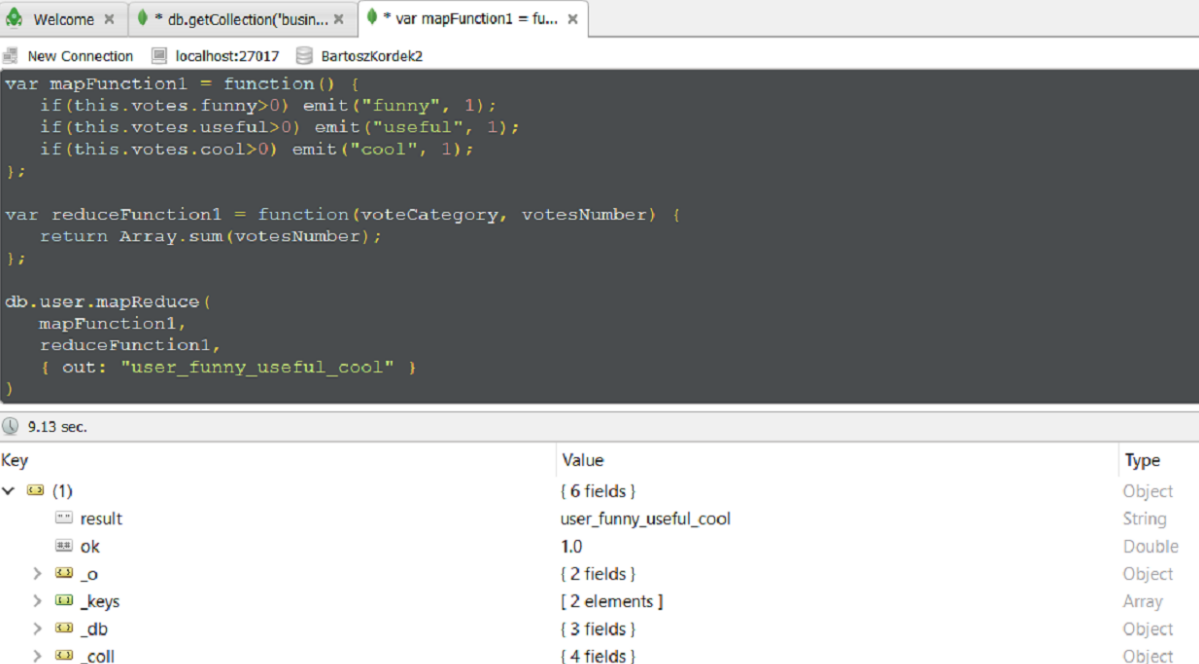
```
db.getCollection('business').aggregate([
  { $match:
    { $and:
      [
        { "categories": "Hotels" },
        { "attributes.Wi-Fi": "free" },
        { "stars": { $gte: 4.5 } }
      ]
    }
  },
  { $group: { _id: { state: "$state"}, total: { $sum: 1 } } }
])
```

business 0.037 sec.

Wyniki:

Key	Value	Type
✓ (1) { 1 field }	{ 2 fields }	Object
✓ _id	{ 1 field }	Object
state	EDH	String
total	13.0	Double
✓ (2) { 1 field }	{ 2 fields }	Object
✓ _id	{ 1 field }	Object
state	MLN	String
total	1.0	Double
✓ (3) { 1 field }	{ 2 fields }	Object
✓ _id	{ 1 field }	Object
state	ON	String
total	2.0	Double
✓ (4) { 1 field }	{ 2 fields }	Object
✓ _id	{ 1 field }	Object
state	WI	String
total	10.0	Double
✓ (5) { 1 field }	{ 2 fields }	Object
✓ _id	{ 1 field }	Object
state	NV	String
total	10.0	Double
✓ (6) { 1 field }	{ 2 fields }	Object
✓ _id	{ 1 field }	Object
state	AZ	String
total	33.0	Double

- d) zwróć, ile recenzji posiadają oceny z każdej kategorii: funny, cool, useful. Przypisanie recenzji do kategorii oznacza, że przynajmniej jedna osoba zagłosowała na recenzje w tej kategorii). Wykorzystaniem mechanizmu map-reduce.



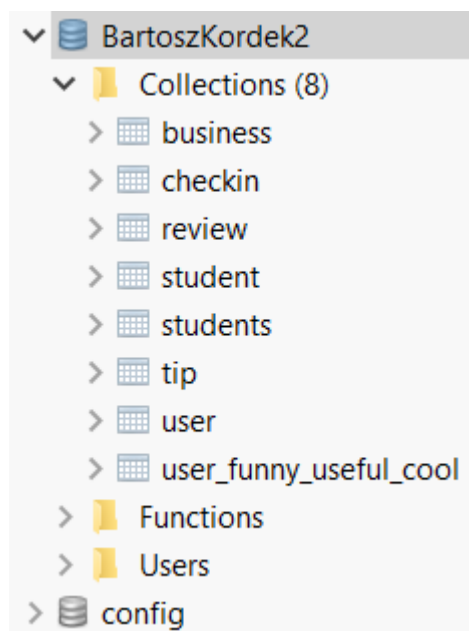
The screenshot shows the MongoDB Shell interface. At the top, there are tabs for 'Welcome', 'db.getCollection("busin...",', and 'var mapFunction1 = fu...'. Below the tabs, the command prompt shows 'localhost:27017 BartoszKordek2'. The main area contains the following JavaScript code:

```
var mapFunction1 = function() {  
  if(this.votes.funny>0) emit("funny", 1);  
  if(this.votes.useful>0) emit("useful", 1);  
  if(this.votes.cool>0) emit("cool", 1);  
};  
  
var reduceFunction1 = function(voteCategory, votesNumber) {  
  return Array.sum(votesNumber);  
};  
  
db.user.mapReduce(  
  mapFunction1,  
  reduceFunction1,  
  { out: "user_funny_useful_cool" }  
)
```

Below the code, the execution time is shown as '9.13 sec.'. The result is displayed in a table with three columns: 'Key', 'Value', and 'Type'.

Key	Value	Type
✓ (1)	{ 6 fields }	Object
result	user_funny_useful_cool	String
ok	1.0	Double
> (3) _o	{ 2 fields }	Object
> (3) _keys	[2 elements]	Array
> (3) _db	{ 3 fields }	Object
> (3) _coll	{ 4 fields }	Object

Została utworzona kolekcja user_funny_useful_cool



Wyniki:

```
db.getCollection('user_funny_useful_cool').find({})
```

user_funny_useful_cool 0.003 sec.

Key	Value	Type
▼ (1) cool	{ 2 fields }	Object
_id	cool	String
value	159951.0	Double
▼ (2) funny	{ 2 fields }	Object
_id	funny	String
value	153130.0	Double
▼ (3) useful	{ 2 fields }	Object
_id	useful	String
value	211034.0	Double

6) Wykonaj zadania punktu 5 z poziomu języka Java:

- wykorzystaj szkielet projektu mongo-lab za pomocą IDE Eclipse
- każde z zadań wykonaj z oddzielnej metody.

a)

```
private long get5StarBusinessCounter() {  
    DBCollection gettedCollection = db.getCollection("business");  
    DBObject query = new BasicDBObject();  
    query.put("stars", 5.0);  
    return gettedCollection.count(query);  
}
```

Wynik:

```
256  
257 public static void main(String args[]) throws UnknownHostException {  
258     MongoLab mongoLab = new MongoLab();  
259     mongoLab.showCollections();  
260     System.out.println("FIVE-STAR BUSINESSES: "+mongoLab.get5StarBusinessCounter());  
261 }  
262  
263
```

Console Problems Javadoc Declaration Coverage

<terminated> MongoLab (2) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (7 lis 2020, 21:58:39)

lis 07, 2020 9:58:41 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster created with settings {hosts=[127.0.0.1:27017], mode=SINGLE, requiredClusterType=
lis 07, 2020 9:58:41 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
lis 07, 2020 9:58:41 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:236}] to 127.0.0.1:27017
lis 07, 2020 9:58:41 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address
lis 07, 2020 9:58:41 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:237}] to 127.0.0.1:27017
collection: business
collection: checkin
collection: review
collection: student
collection: students
collection: tip
collection: user
collection: user_funny_useful_cool
FIVE-STAR BUSINESSES: 5097

Jak widać powyżej, uzyskałem taki sam wynik jak przy pomocy terminala Robo 3T

- b) ilość restauracji w każdym mieście, wynik posortuj malejąco na podstawie liczby.
Pole categories w dokumencie business musi zawierać wartość Restaurants.

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

    MongoCollection<Document> gettedCollection = mdb.getCollection("business");

    Document match = new Document("$match", new Document("categories", "Restaurants"));

    Document groupCity = new Document("_id", "$city");
    groupCity.put("quantity", new Document("$sum", 1));
    Document group = new Document("$group", groupCity);

    Document sort = new Document("$sort", new Document("quantity", -1));

    List<Document> pipeline = Arrays.asList(match, group, sort);
    return gettedCollection.aggregate(pipeline);
}
```

Wynik:

```
257 public static void main(String args[]) throws UnknownHostException {
258     MongoLab mongoLab = new MongoLab();
259     mongoLab.showCollections();
260     System.out.println("RESTAURANTS BY CITY");
261     System.out.println(JSON.serialize(mongoLab.getRestaurantsByCity()));
262 }
263
264
```

Console Problems Javadoc Declaration Coverage

<terminated> MongoLab (2) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (7 lis 2020, 22:16:01)

collection: students
collection: tip
collection: user
collection: user_funny_useful_cool
RESTAURANTS BY CITY
[{ "_id": "Las Vegas", "quantity": 3855 }, { "_id": "Phoenix", "quantity": 2493 }, { "_id": "Edinburgh", "quantity": 1049 }, { "_id": "Scottsdale", "quantity": 1023 }, { "_id": "Mesa", "quantity": 693 }, { "_id": "Madison", "quantity": 679 }, { "_id": "Tempe", "quantity": 672 }, { "_id": "Henderson", "quantity": 564 }, { "_id": "Chandler", "quantity": 548 }, { "_id": "Glendale", "quantity": 422 }, { "_id": "Gilbert", "quantity": 317 }, { "_id": "Peoria", "quantity": 221 }, { "_id": "North Las Vegas", "quantity": 198 }, { "_id": "Surprise", "quantity": 144 }, { "_id": "Goodyear", "quantity": 119 }, { "_id": "Waterloo", "quantity": 117 }, { "_id": "Avondale", "quantity": 100 }, { "_id": "Kitchener", "quantity": 96 }, { "_id": "Queen Creek", "quantity": 82 }, { "_id": "Middleton", "quantity": 66 }, { "_id": "Cave Creek", "quantity": 63 }, { "_id": "Casa Grande", "quantity": 61 }, { "_id": "Fountain Hills", "quantity": 47 }, { "_id": "Apache Junction", "quantity": 44 }, { "_id": "Buckeye", "quantity": 42 }, { "_id": "Sun Prairie", "quantity": 39 }, { "_id": "Fitchburg", "quantity": 38 }, { "_id": "Maricopa", "quantity": 37 }, { "_id": "Monona", "quantity": 32 }, { "_id": "Sun City", "quantity": 31 }, { "_id": "Wickenburg", "quantity": 31 }, { "_id": "Litchfield Park", "quantity": 27 }, { "_id": "Paradise Valley", "quantity": 26 }, { "_id": "Laveen", "quantity": 25 }, { "_id": "Anthem", "quantity": 24 }, { "_id": "Verona", "quantity": 19 }, { "_id": "Carefree", "quantity": 16 }, { "_id": "San Tan Valley", "quantity": 16 }, { "_id": "Tolleson", "quantity": 15 }, { "_id": "Gold Canyon", "quantity": 15 }, { "_id": "Waunakee", "quantity": 14 }, { "_id": "El Mirage", "quantity": 14 }, { "_id": "Gila Bend", "quantity": 9 }, { "_id": "Sun City West", "quantity": 9 }, { "_id": "Boulder City", "quantity": 7 }, { "_id": "Florence", "quantity": 7 }, { "_id": "N Las Vegas", "quantity": 7 }, { "_id": "Cottage Grove", "quantity": 6 }, { "_id": "Coolidge", "quantity": 6 }, { "_id": "Paradise", "quantity": 6 }, { "_id": "De Forest", "quantity": 5 }, {

```
[ { "_id": "Las Vegas", "quantity": 3855 }, { "_id": "Phoenix", "quantity": 2493 }, { "_id": "Edinburgh", "quantity": 1049 }, { "_id": "Scottsdale", "quantity": 1023 }, { "_id": "Mesa", "quantity": 693 }, { "_id": "Madison", "quantity": 679 }, { "_id": "Tempe", "quantity": 672 }, { "_id": "Henderson", "quantity": 564 }, { "_id": "Chandler", "quantity": 548 }, { "_id": "Glendale", "quantity": 422 }, { "_id": "Gilbert", "quantity": 317 }, { "_id": "Peoria", "quantity": 221 }, { "_id": "North Las Vegas", "quantity": 198 }, { "_id": "Surprise", "quantity": 144 }, { "_id": "Goodyear", "quantity": 119 }, { "_id": "Waterloo", "quantity": 117 }, { "_id": "Avondale", "quantity": 100 }, { "_id": "Kitchener", "quantity": 96 }, { "_id": "Queen Creek", "quantity": 82 }, { "_id": "Middleton", "quantity": 66 }, { "_id": "Cave Creek", "quantity": 63 }, { "_id": "Casa Grande", "quantity": 61 }, { "_id": "Fountain Hills", "quantity": 47 }, { "_id": "Apache Junction", "quantity": 44 }, { "_id": "Buckeye", "quantity": 42 }, { "_id": "Sun Prairie", "quantity": 39 }, { "_id": "Fitchburg", "quantity": 38 }, { "_id": "Maricopa", "quantity": 37 }, { "_id": "Monona", "quantity": 32 }, { "_id": "Sun City", "quantity": 31 }, { "_id": "Wickenburg", "quantity": 31 }, { "_id": "Litchfield Park", "quantity": 27 }, { "_id": "Paradise Valley", "quantity": 26 }, { "_id": "Laveen", "quantity": 25 }, { "_id": "Anthem", "quantity": 24 }, { "_id": "Verona", "quantity": 19 }, { "_id": "Carefree", "quantity": 16 }, { "_id": "San Tan Valley", "quantity": 16 }, { "_id": "Tolleson", "quantity": 15 }, { "_id": "Gold Canyon", "quantity": 15 }, { "_id": "Waunakee", "quantity": 14 }, { "_id": "El Mirage", "quantity": 14 }, { "_id": "Gila Bend", "quantity": 9 }, { "_id": "Sun City West", "quantity": 9 }, { "_id": "Boulder City", "quantity": 7 }, { "_id": "Florence", "quantity": 7 }, { "_id": "N Las Vegas", "quantity": 7 }, { "_id": "Cottage Grove", "quantity": 6 }, { "_id": "Coolidge", "quantity": 6 }, { "_id": "Paradise", "quantity": 6 }, { "_id": "De Forest", "quantity": 5 }, {
```

```

"_id": "Guadalupe", "quantity": 5}, {"_id": "Spring Valley", "quantity": 5}, {"_id":
"Musselburgh", "quantity": 5}, {"_id": "Mc Farland", "quantity": 5}, {"_id": "Sun Lakes",
"quantity": 4}, {"_id": "New River", "quantity": 4}, {"_id": "Youngtown", "quantity": 4}, {"
_id": "South Queensferry", "quantity": 4}, {"_id": "Windsor", "quantity": 3}, {"_id": "Fort
McDowell", "quantity": 3}, {"_id": "Enterprise", "quantity": 3}, {"_id": "McFarland",
"quantity": 3}, {"_id": "Ahwatukee", "quantity": 3}, {"_id": "DeForest", "quantity": 3}, {"
_id": "Morristown", "quantity": 2}, {"_id": "Stoughton", "quantity": 2}, {"_id": "N. Las
Vegas", "quantity": 2}, {"_id": "Pheonix", "quantity": 2}, {"_id": "Tonopah", "quantity": 2}
, {"_id": "Dalkeith", "quantity": 2}, {"_id": "Cambridge", "quantity": 2}, {"_id": "New
Town", "quantity": 2}, {"_id": "Queensferry", "quantity": 2}, {"_id": "Central City Village",
"quantity": 2}, {"_id": "St Clements", "quantity": 1}, {"_id": "Glendale Az", "quantity": 1},
{"_id": "Old Town", "quantity": 1}, {"_id": "Black Canyon City", "quantity": 1}, {"_id":
"Tortilla Flat", "quantity": 1}, {"_id": "Clark County", "quantity": 1}, {"_id": "Juniper
Green", "quantity": 1}, {"_id": "Rio Verde", "quantity": 1}, {"_id": "Dane", "quantity": 1},
{"_id": "Green Valley", "quantity": 1}, {"_id": "Higley", "quantity": 1}, {"_id": "Las Vegas
", "quantity": 1}, {"_id": "Sedona", "quantity": 1}, {"_id": "North Scottsdale", "quantity":
1}, {"_id": "Nellis Afb", "quantity": 1}, {"_id": "Summerlin South", "quantity": 1}, {"_id":
"St Jacobs", "quantity": 1}, {"_id": "Fort Mcdowell", "quantity": 1}, {"_id": "NELLIS AFB",
"quantity": 1}, {"_id": "Trempealeau", "quantity": 1}, {"_id": "Woolwich", "quantity": 1},
{"_id": "Nellis AFB", "quantity": 1}, {"_id": "Ratho", "quantity": 1}, {"_id": "Loanhead",
"quantity": 1}, {"_id": "South Gyle", "quantity": 1}, {"_id": "Summerlin", "quantity": 1}, {"
_id": "Atlanta", "quantity": 1}, {"_id": "Lasswade", "quantity": 1}, {"_id": "City of
Edinburgh", "quantity": 1}, {"_id": "Inverkeithing", "quantity": 1}, {"_id": "Saint Jacobs",
"quantity": 1}, {"_id": "Phoenix Sky Harbor Center", "quantity": 1}, {"_id": "Stockbridge",
"quantity": 1}]

```

Otrzymano takie same wyniki jak w przypadku wykonania zadania w terminalu aplikacji Robo 3T.

- c) ilość hoteli (atrybut categories powinien mieć wartość Hotels) w każdym stanie/okręgu (state), które posiadają darmowe Wi-fi (pole attributes, klucz-wartość 'Wi-Fi': 'free') oraz ocenę co najmniej 4.5 gwiazdki. Wykorzystaj funkcję group.

```
public AggregateIterable<Document> getHotelAmount(){
    MongoClient<Document> gettedCollection = mdb.getCollection("business");

    Document category = new Document("categories", "Hotels");
    Document wifi = new Document("attributes.Wi-Fi", "free");
    Document stars = new Document("stars", new Document("$gte", 4.5));

    Document match = new Document("$match", new Document("$and", Arrays.asList(category, wifi, stars)));

    Document groupByFields = new Document("_id", "$state");
    groupByFields.put("quantity", new Document("$sum", 1));
    Document group = new Document("$group", groupByFields);

    List<Document> pipeline = Arrays.asList(match, group);

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

Wyniki:

```
320 public static void main(String args[]) throws UnknownHostException {
321     MongoLab mongolab = new MongoLab();
322     mongolab.showCollections();
323     System.out.println(JSON.serialize(mongolab.getHotelAmountReworked()));
324 }
325
326
```

<terminated> MongoLab (2) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (8 lis 2020, 11:49:53)

collection: business
collection: checkin
collection: review
collection: student
collection: students
collection: tip
collection: user
collection: user_funny_useful_cool

[{ "_id" : "EDH", "quantity" : 13 }, { "_id" : "MLN", "quantity" : 1 }, { "_id" : "ON", "quantity" : 2 }, { "_id" : "WI", "quantity" : 10 }, { "_id" : "NV", "quantity" : 10 }, { "_id" : "AZ", "quantity" : 33 }]

Otrzymano identyczne wyniki jak w przypadku użycia terminala Robo 3T. Ważne jest użycie operatora "and".

- d) zwróć, ile recenzji posiadają oceny z każdej kategorii: funny, cool, useful. Przypisanie recenzji do kategorii oznacza, że przynajmniej jedna osoba zagłosowała na recenzję w tej kategorii). Wykorzystaniem mechanizmu map-reduce.

```
private void getVotesCounts(){
    DBCollection collection = db.getCollection("user");

    String map = "function() {\r\n" +
        "    if(this.votes.funny>0) emit(\"funny\", 1);\r\n" +
        "    if(this.votes.useful>0) emit(\"useful\", 1);\r\n" +
        "    if(this.votes.cool>0) emit(\"cool\", 1);\r\n" +
        "};";

    String reduce = "function(voteCategory, votesNumber) {\r\n" +
        "    return Array.sum(votesNumber);\r\n" +
        "};";

    MapReduceCommand cmd = new MapReduceCommand(collection, map, reduce, null, MapReduceCommand.OutputType.INLINE, null);
    MapReduceOutput out = collection.mapReduce(cmd);
    for(DBObject o : out.results()) {
        System.out.println(o.toString());
    }
}
```

Wyniki:

```
255
256 public static void main(String args[]) throws UnknownHostException {
257     MongoLab mongoLab = new MongoLab();
258     mongoLab.showCollections();
259     System.out.println("VOTES COUNTER");
260     mongoLab.getVotesCounts();
261 }
262
```

<terminated> MongoLab (2) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (7 lis 2021)

collection: student
collection: students
collection: tip
collection: user
collection: user_funny_useful_cool
VOTES COUNTER
{ "_id": "cool", "value": 159951.0}
{ "_id": "funny", "value": 153130.0}
{ "_id": "useful", "value": 211034.0}

Posłużyłem się tak jak poprzednio kolekcją "user". Wyniki takie same jak w przypadku skorzystania z terminala Robo 3T.

7) Napisz kod w języku Java (metoda), który zwróci użytkownika (nazwa użytkownika) o największej liczbie pozytywnych recenzji (ocena co najmniej 4.5).

```
private String getMostPositiveVotesUser(){

    MongoCollection<Document> gettedCollection = mdb.getCollection("user");

    Document query = new Document("average_stars", new Document("$gt", 4.5));

    List<Document> pipeline = Arrays.asList(query);
    FindIterable<Document> documents = gettedCollection.find(query);

    MongoCursor<Document> cursor = documents.iterator();

    int reviewMax = 0;
    String mostPositiveVotesUser = null;
    Document currentDocument = null;

    while(cursor.hasNext()) {
        currentDocument = cursor.next();
        if(currentDocument.getInteger("review_count")>reviewMax) {
            reviewMax = currentDocument.getInteger("review_count");
            mostPositiveVotesUser = currentDocument.getString("name");
        }
    }

    return mostPositiveVotesUser;
}
```

Szybsze rozwiązanie z użyciem agregatora. Wynikiem komendy na bazie danych jest posortowana malejąco lista wg pola "review_count". Przy pomocy funkcji javowej został wybrany pierwszy z elementów listy.

```
private String getMostPositiveVotesUserFasterSolution(){

    MongoCollection<Document> gettedCollection = mdb.getCollection("user");

    Document filter = new Document("average_stars", new Document("$gt", 4.5));
    Document match = new Document("$match", filter);

    Document sort = new Document("$sort", new Document("review_count", -1));

    List<Document> pipeline = Arrays.asList(match, sort);
    AggregateIterable<Document> documents = gettedCollection.aggregate(pipeline);

    return documents.iterator().next().getString("name");
}
```

```

338 public static void main(String args[]) throws UnknownHostException {
339     MongoLab mongoLab = new MongoLab();
340     mongoLab.showCollections();
341     startTime = System.nanoTime();
342     System.out.print("FIRST SOLUTION: Result: "+mongoLab.getMostPositiveVotesUser());
343     endTime = System.nanoTime();
344     System.out.println(" Execution Time: "+(endTime-startTime)+" [ns]");
345     startTime = System.nanoTime();
346     System.out.print("FASTER SOLUTION: Result: "+mongoLab.getMostPositiveVotesUserFasterSolution());
347     endTime = System.nanoTime();
348     System.out.println(" Execution Time: "+(endTime-startTime)+" [ns]");
349 }
350
351

```

Console Problems Javadoc Declaration Coverage

<terminated> MongoLab (2) [Java Application] C:\Program Files\Java\jdk1.8.0_161\bin\javaw.exe (8 lis 2020, 13:37:13)

```

INFO: Cluster created with settings {hosts=[127.0.0.1:27017], mode=SINGLE, requiredClusterType=UNKNOWN, server
lis 08, 2020 1:37:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Cluster description not yet available. Waiting for 30000 ms before timing out
lis 08, 2020 1:37:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:1, serverValue:7}] to 127.0.0.1:27017
lis 08, 2020 1:37:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Monitor thread successfully connected to server with description ServerDescription{address=127.0.0.1:270
lis 08, 2020 1:37:15 PM com.mongodb.diagnostics.logging.JULLogger log
INFO: Opened connection [connectionId{localValue:2, serverValue:8}] to 127.0.0.1:27017
collection: business
collection: checkin
collection: review
collection: student
collection: students
collection: tip
collection: user
collection: user_funny_useful_cool
FIRST SOLUTION: Result: Brian Execution Time: 1273904400 [ns]
FASTER SOLUTION: Result: Brian Execution Time: 405593000 [ns]

```

Jak widać w obu przypadkach wynik jest ten sam, lecz przy użyciu drugiego sposobu, czas wykonania jest dużo krótszy.