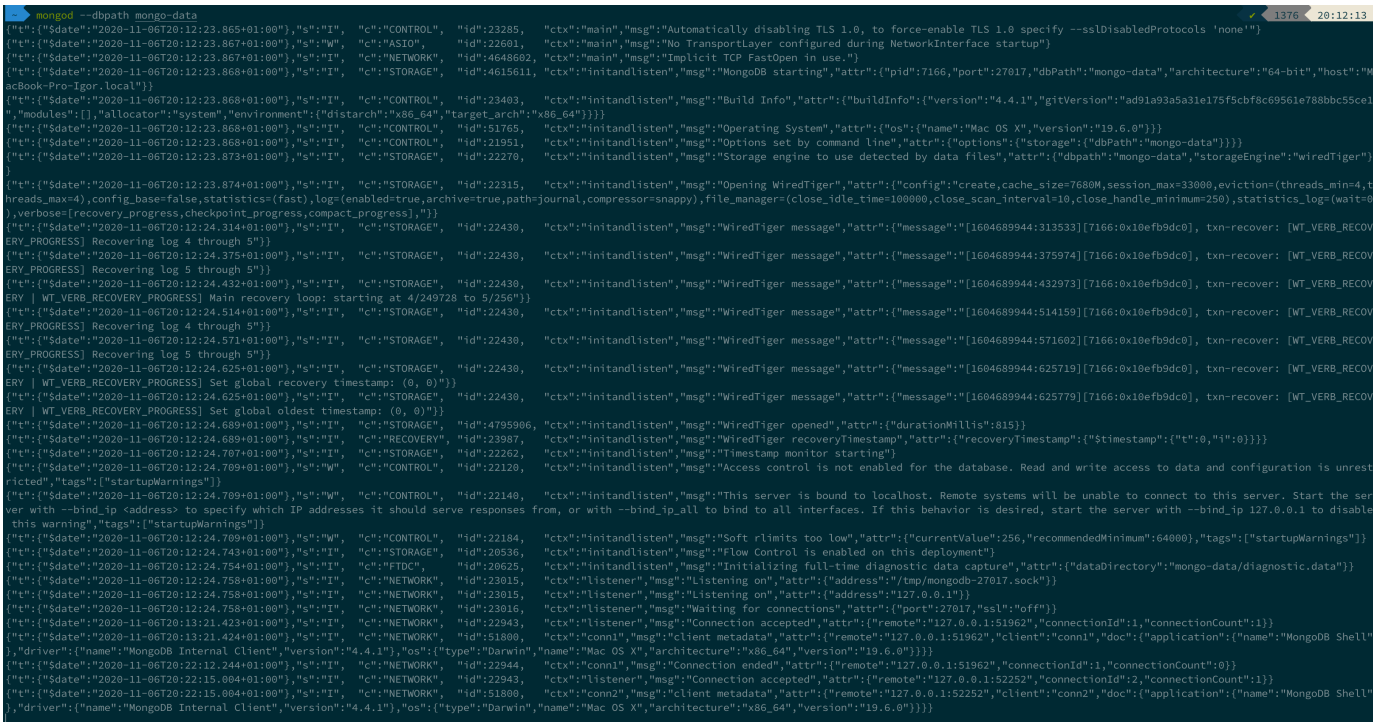
Laboratorium 3: MongoDB

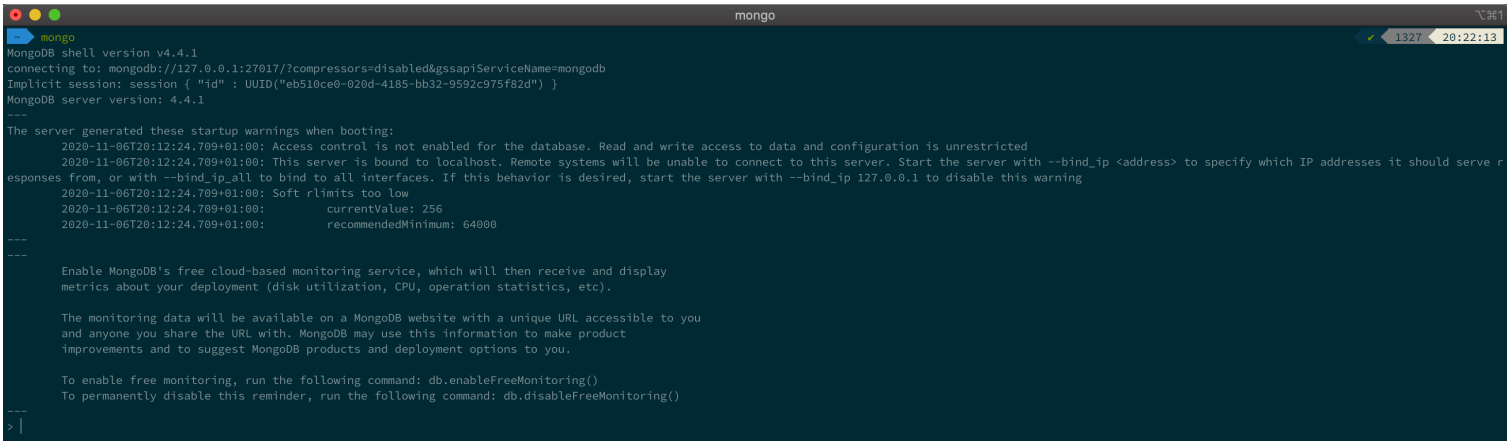
Igor Dzierwa
Grupa: 1

1.Uruchom usługę MongoDB:

-> **`mongod`** — głównym procesem demona systemu MongoDB.



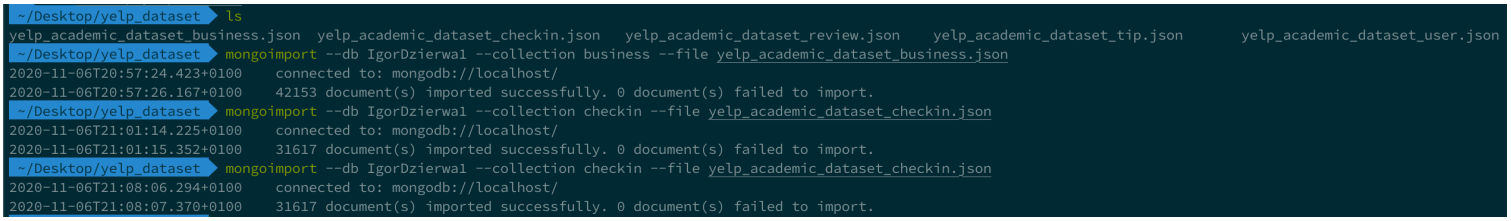
-> **`mongo`** — interaktywny interfejs powłoki JavaScript.



2.Zaimportuj do MongoDB pliki yelp_academic_data

a) wykorzystaj komendę:

```
mongoimport --db <db-name> --collection <coll- name> --type json --file <file>
```



```
~/Desktop/yelp_dataset  mongoimport --db IgorDzierwa1 --collection review --file yelp_academic_dataset_review.json
2020-11-06T21:08:37.513+0100    connected to: mongodb://localhost/
2020-11-06T21:08:40.516+0100    [##......................] IgorDzierwa1.review  96.0MB/997MB (9.6%)
2020-11-06T21:08:43.515+0100    [####....................] IgorDzierwa1.review  192MB/997MB (19.3%)
2020-11-06T21:08:46.518+0100    [#######.................] IgorDzierwa1.review  297MB/997MB (29.8%)
2020-11-06T21:08:49.514+0100    [#########...............] IgorDzierwa1.review  400MB/997MB (40.1%)
2020-11-06T21:08:52.514+0100    [###########.............] IgorDzierwa1.review  492MB/997MB (49.4%)
2020-11-06T21:08:55.516+0100    [##############..........] IgorDzierwa1.review  591MB/997MB (59.3%)
2020-11-06T21:08:58.517+0100    [################........] IgorDzierwa1.review  683MB/997MB (68.5%)
2020-11-06T21:09:01.517+0100    [##################......] IgorDzierwa1.review  777MB/997MB (77.9%)
2020-11-06T21:09:04.515+0100    [####################....] IgorDzierwa1.review  869MB/997MB (87.1%)
2020-11-06T21:09:07.518+0100    [######################..] IgorDzierwa1.review  964MB/997MB (96.6%)
2020-11-06T21:09:08.711+0100    [########################] IgorDzierwa1.review  997MB/997MB (100.0%)
2020-11-06T21:09:08.711+0100    1125458 document(s) imported successfully. 0 document(s) failed to import.
~/Desktop/yelp_dataset  mongoimport --db IgorDzierwa1 --collection tip --file yelp_academic_dataset_tip.json
2020-11-06T21:09:43.862+0100    connected to: mongodb://localhost/
2020-11-06T21:09:46.865+0100    [#############...........] IgorDzierwa1.tip     43.0MB/75.7MB (56.8%)
2020-11-06T21:09:49.047+0100    [########################] IgorDzierwa1.tip     75.7MB/75.7MB (100.0%)
2020-11-06T21:09:49.047+0100    403210 document(s) imported successfully. 0 document(s) failed to import.
~/Desktop/yelp_dataset  mongoimport --db IgorDzierwa1 --collection user --file yelp_academic_dataset_user.json
2020-11-06T21:10:05.815+0100    connected to: mongodb://localhost/
2020-11-06T21:10:08.815+0100    [##########..............] IgorDzierwa1.user    50.5MB/113MB (44.6%)
2020-11-06T21:10:11.815+0100    [####################....] IgorDzierwa1.user    100MB/113MB (88.5%)
2020-11-06T21:10:12.584+0100    [########################] IgorDzierwa1.user    113MB/113MB (100.0%)
2020-11-06T21:10:12.584+0100    252898 document(s) imported successfully. 0 document(s) failed to import.
```

b) stwórz nową bazę danych o nazwie: `ImieNazwiskoNrGrupy`:

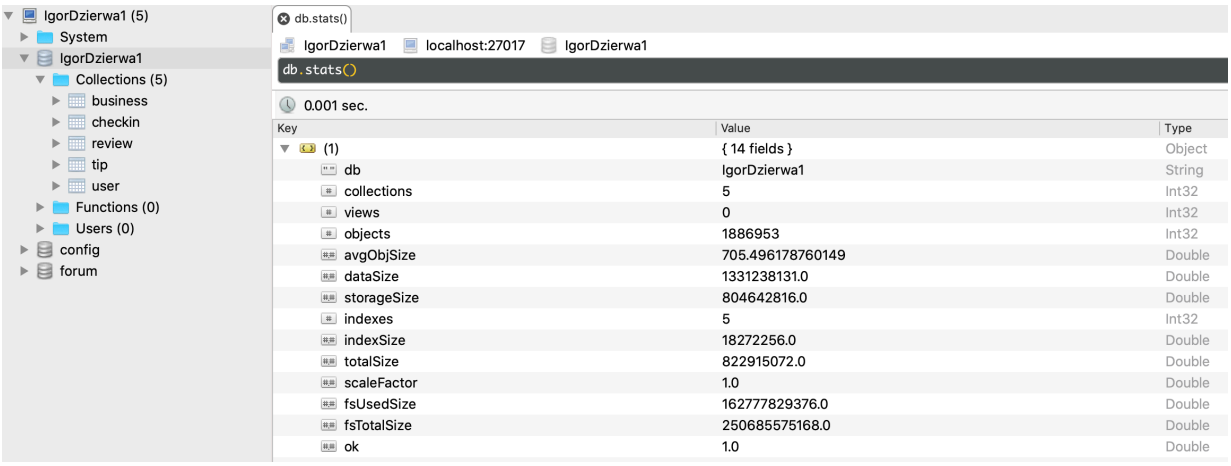
```
> use IgorDzierwa1
switched to db IgorDzierwa1
```

```
> show dbs
IgorDzierwa1  0.766GB
admin         0.000GB
config        0.000GB
forum         0.000GB
local         0.000GB
> use IgorDzierwa1
```

c) Przyporządkuj kolekcjom opartych na importowanych plikach .json odpowiednie nazwy.

```
> show collections
business
checkin
review
tip
user
>
```

3. Połącz się z bazą z użyciem narzędzia Robo3T:

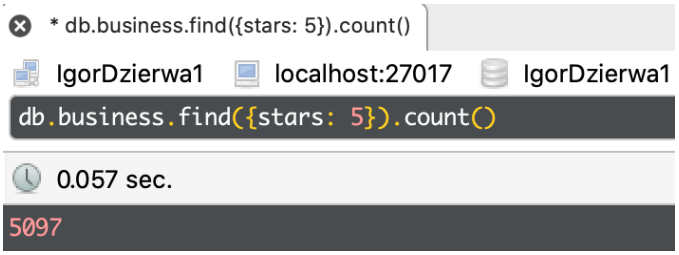


4. Za pomocą narzędzia Robo 3T wykonaj polecenie dodające do stworzonej bazy kolekcję „student”

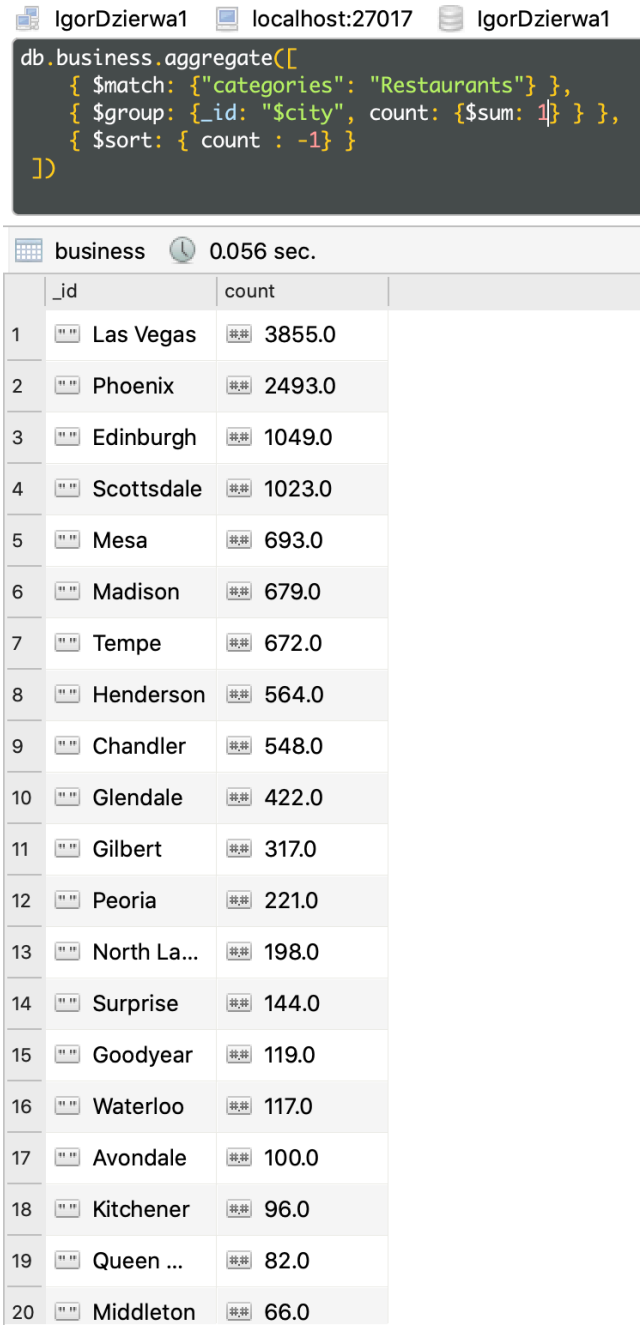
a) wprowadź własne dane do kolekcji: imię, nazwisko, obecność (typ bool), ocena z lab. (null), aktualna data, zaliczone przedmioty *(min 3 przykładowe).*
b) wyświetl wynik dodania danej w formie `.json txt`.

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

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



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



	_id	count
1	Las Vegas	3855.0
2	Phoenix	2493.0
3	Edinburgh	1049.0
4	Scottsdale	1023.0
5	Mesa	693.0
6	Madison	679.0
7	Tempe	672.0
8	Henderson	564.0
9	Chandler	548.0
10	Glendale	422.0
11	Gilbert	317.0
12	Peoria	221.0
13	North La...	198.0
14	Surprise	144.0
15	Goodyear	119.0
16	Waterloo	117.0
17	Avondale	100.0
18	Kitchener	96.0
19	Queen ...	82.0
20	Middleton	66.0

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.

IgorDzierwa1 localhost:27017 IgorDzierwa1

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

business 0.042 sec.

```
/* 1 */
{
    "_id" : "NV",
    "count" : 10.0
}

/* 2 */
{
    "_id" : "EDH",
    "count" : 13.0
}

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

/* 4 */
{
    "_id" : "AZ",
    "count" : 33.0
}

/* 5 */
{
    "_id" : "WI",
    "count" : 10.0
}

/* 6 */
{
    "_id" : "ON",
    "count" : 2.0
}
```

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.

```
//funkcja mapujaca
var map = function() {
    emit("votes", this.votes);
};
//funkcja redukujaca
var reduce = function(voteName, array) {
    var valueFunny = 0;
    var valueCool = 0;
    var valueUseful = 0;

    for(var i = 0; i < array.length; i++) {
        if(array[i].funny > 0)
            valueFunny += 1;
        if(array[i].cool > 0 )
            valueCool += 1;
        if(array[i].useful > 0)
            valueUseful += 1;
    }
    return {funny: valueFunny, cool: valueCool, useful: valueUseful};
 };

db.user.mapReduce(
    map,
    reduce,
    {
        out: "votes_summary"
    }
   )

db.votes_summary.find({})
```

3.07 sec.

```
/* 1 */
{
    "result" : "votes_summary",
    "ok" : 1.0,
    "_o" : {
        "result" : "votes_summary",
        "ok" : 1.0
    },
    "_keys" : [
        "result",
        "ok"
    ],
    "_db" : {
```

votes_summary 0.142 sec.

```
/* 1 */
{
    "_id" : "votes",
    "value" : {
        "funny" : 153130.0,
        "cool" : 159951.0,
        "useful" : 211034.0
    }
}
```

6. Wykonaj zadania punktu 4 z poziomu języka Java:

*Pobrane pliki JAR:

- mongo-java-driver-3.12.7.jar
- mongodb-driver-core-4.1.1.jar

```java
import com.mongodb.BasicDBObject;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.MongoClient;
import org.bson.Document;

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

    public Mongo(String hostname, int port, String databaseName) {
        this.mongoClient = new MongoClient(hostname , port);
        this.db = mongoClient.getDatabase(databaseName);
    }

    public MongoClient getMongoClient() {
        return this.mongoClient;
    }

    public MongoDatabase getDb() {
        return this.db;
    }

    public void showCollections() {
        for (String name : db.listCollectionNames()) {
            System.out.println("colection name: " + name);
        }
    }
```

a) Odwzorowanie punktu 5a:

```java
public void exercise5a() {
    ArrayList<Document> results = new ArrayList<>();
    MongoCollection<Document> collection = db.getCollection( s: "business");
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("stars", 5);
    collection.find(whereQuery).into(results);
    System.out.println("Count: " + results.size());
}
```

```
Count: 5097
```

b) Odwzorowanie punktu 5b:

```java
public void exercise5b() {
    MongoCollection<Document> collection = db.getCollection( s: "business");
    List<Bson> results = Arrays.asList(
            Aggregates.match(Filters.eq( fieldName: "categories",  value: "Restaurants")),
            Aggregates.group( id: "$city", Accumulators.sum( fieldName: "count",  expression: 1)),
            Aggregates.sort(Sorts.descending( ...fieldNames: "count"))
    );

    AggregateIterable iterable = collection.aggregate(results);

    for(Object i : iterable) {
        System.out.println(i);
    }
}
```

- Fragment zwróconej zawartości:

```
Document{{_id=Las Vegas, count=3855}}
Document{{_id=Phoenix, count=2493}}
Document{{_id=Edinburgh, count=1049}}
Document{{_id=Scottsdale, count=1023}}
Document{{_id=Mesa, count=693}}
Document{{_id=Madison, count=679}}
Document{{_id=Tempe, count=672}}
Document{{_id=Henderson, count=564}}
Document{{_id=Chandler, count=548}}
Document{{_id=Glendale, count=422}}
Document{{_id=Gilbert, count=317}}
Document{{_id=Peoria, count=221}}
Document{{_id=North Las Vegas, count=198}}
Document{{_id=Surprise, count=144}}
Document{{_id=Goodyear, count=119}}
Document{{_id=Waterloo, count=117}}
Document{{_id=Avondale, count=100}}
Document{{_id=Kitchener, count=96}}
Document{{_id=Queen Creek, count=82}}
Document{{_id=Middleton, count=66}}
Document{{_id=Cave Creek, count=63}}
Document{{_id=Casa Grande, count=61}}
Document{{_id=Fountain Hills, count=47}}
Document{{_id=Apache Junction, count=44}}
Document{{_id=Buckeye, count=42}}
Document{{_id=Sun Prairie, count=39}}
Document{{_id=Fitchburg, count=38}}
Document{{_id=Maricopa, count=37}}
Document{{_id=Monona, count=32}}
Document{{_id=Sun City, count=31}}
Document{{_id=Wickenburg, count=31}}
Document{{_id=Litchfield Park, count=27}}
Document{{_id=Paradise Valley, count=26}}
Document{{_id=Laveen, count=25}}
Document{{_id=Anthem, count=24}}
Document{{_id=Verona, count=19}}
Document{{_id=San Tan Valley, count=16}}
```

c) Odwzorowanie punktu 5c:

```
public void exercise5c() {
    MongoCollection<Document> collection = db.getCollection( s: "business");
    List<Bson> results = Arrays.asList(
            Aggregates.match(Filters.and(
                    Filters.eq( fieldName: "categories",  value: "Hotels"),
                    Filters.eq( fieldName: "attributes.Wi-Fi",  value: "free"),
                    Filters.gte( fieldName: "stars",  value: 4.5))),
            Aggregates.group( id: "$state", Accumulators.sum( fieldName: "count",  expression: 1)));

    AggregateIterable iterable = collection.aggregate(results);

    for(Object i : iterable) {
        System.out.println(i);
    }
}
```

```
Document{{_id=NV, count=10}}
Document{{_id=EDH, count=13}}
Document{{_id=MLN, count=1}}
Document{{_id=AZ, count=33}}
Document{{_id=WI, count=10}}
Document{{_id=ON, count=2}}
```

d) Odwzorowanie punktu 5d:

```java
public void exercise5d() {
    ArrayList<Document> results = new ArrayList<>();
    Map<String, Integer> counter = new HashMap<>();
    int funnyCount = 0;
    int coolCount = 0;
    int usefulCount = 0;
    MongoCollection<Document> collection = db.getCollection( s: "user");

    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("votes", new Document("$exists", true));

    collection.find(whereQuery).into(results);

    for(Document user : results) {
        Document votes = (Document) user.get("votes");
        if(votes.getInteger( key: "funny") > 0)
            funnyCount++;
        if(votes.getInteger( key: "cool") > 0)
            coolCount++;
        if(votes.getInteger( key: "useful") > 0)
            usefulCount++;
    }

    counter.put("Funny", funnyCount);
    counter.put("Cool", coolCount);
    counter.put("Useful", usefulCount);

    System.out.println(counter);
}
```

```
{Cool=159951, Useful=211034, Funny=153130}
```

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

```java
public void exercise7() {
    MongoCollection<Document> collection = db.getCollection( s: "user");

    List<Bson> results = Arrays.asList(
            Aggregates.match(Filters.gte( fieldName: "average_stars", value: 4.5)),
            Aggregates.sort(Sorts.descending( ...fieldNames: "review_count")
            ));

    AggregateIterable iterable = collection.aggregate(results);
    Document user = (Document) iterable.first();
    System.out.println("Name: " + user.getString( key: "name") +
            "\n" +
            "Review count: " + user.getInteger( key: "review_count") +
            "\n" +
            "Average rate: " + user.getDouble( key: "average_stars"));
}
```