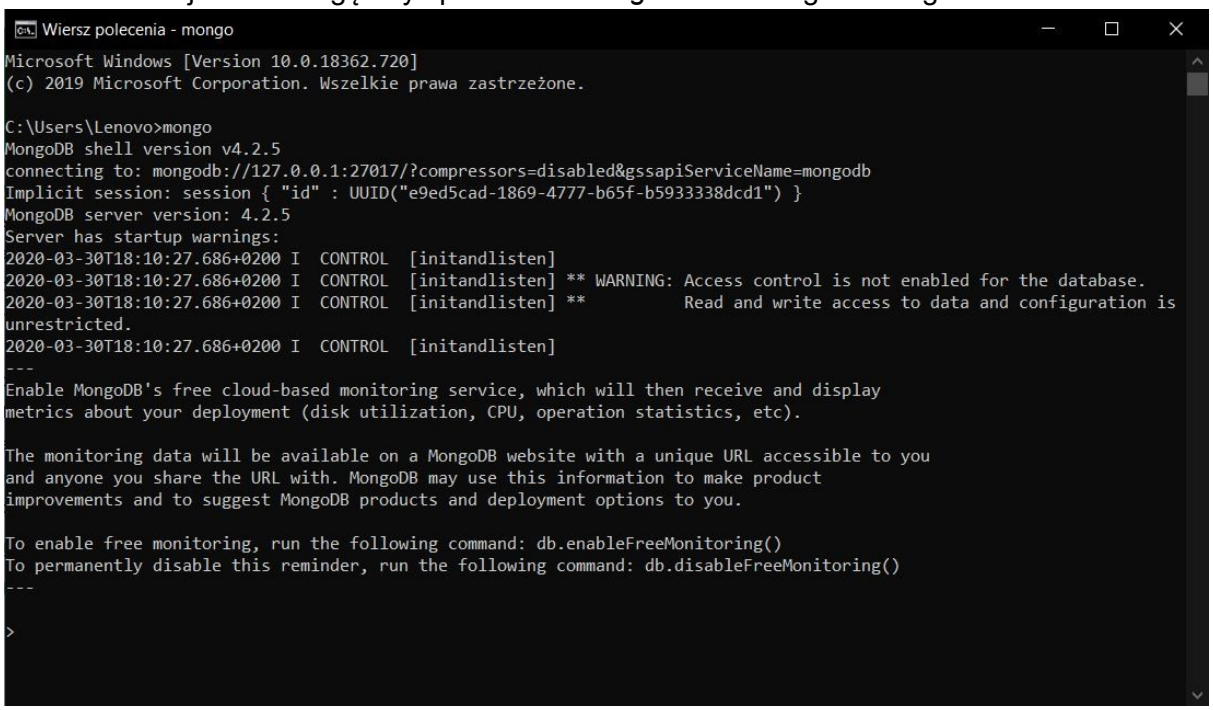
MongoDB

Bartosz Szar

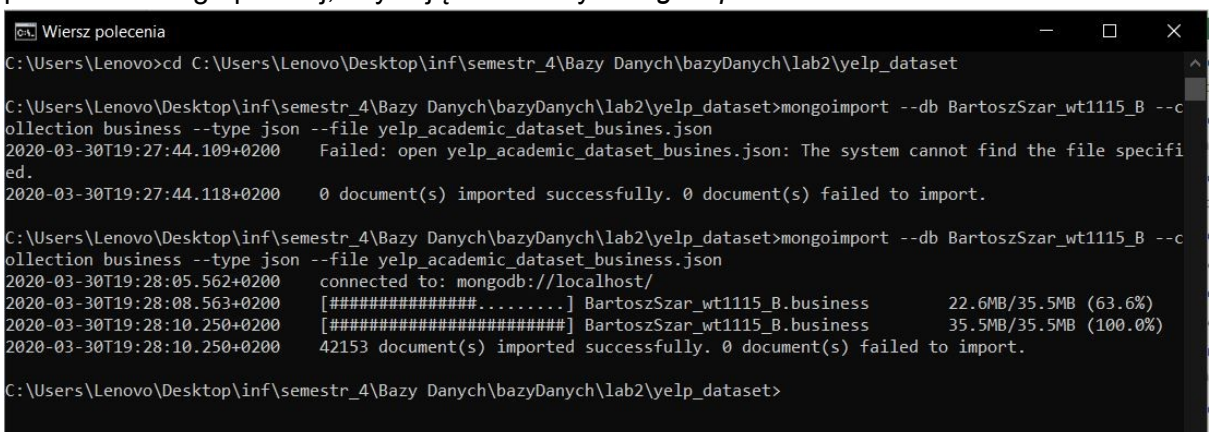
zad1

Po zainstalowaniu oprogramowania i dodaniu odpowiedniej ścieżki do zmiennej środowiskowej PATH mogę użyć polecenia *mongo* z dowolnego katalogu:



zad2

Każdy z 5 plików *JSON* zaimportowałem do swojej bazy w sposób analogiczny do przedstawionego poniżej, używając komendy *mongoimport*:

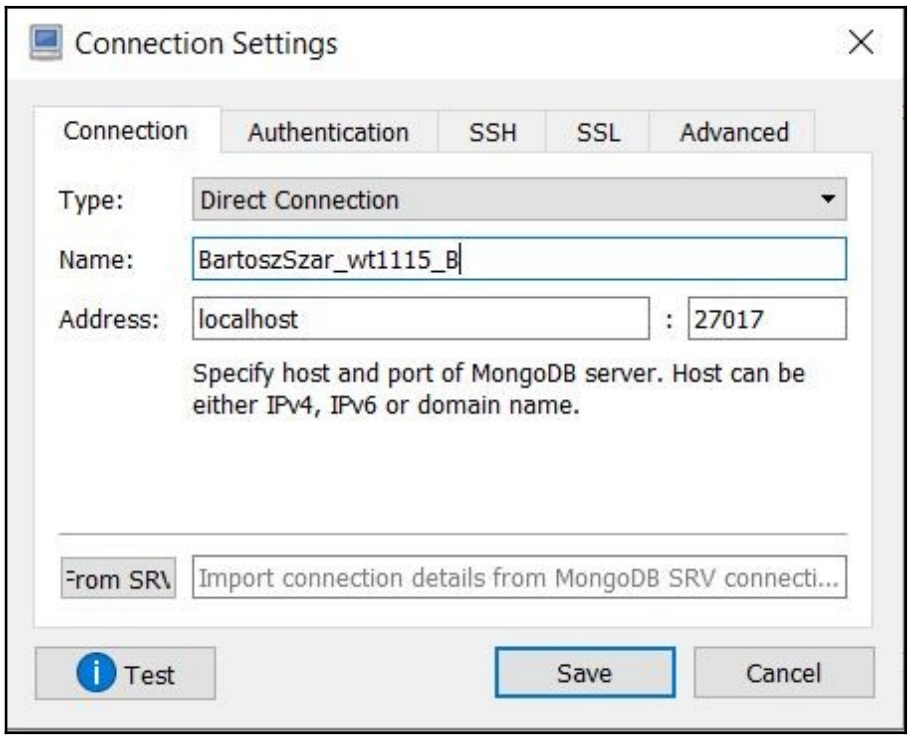


Po wykonaniu importu sprawdziłem zawartość kolekcji w bazie:

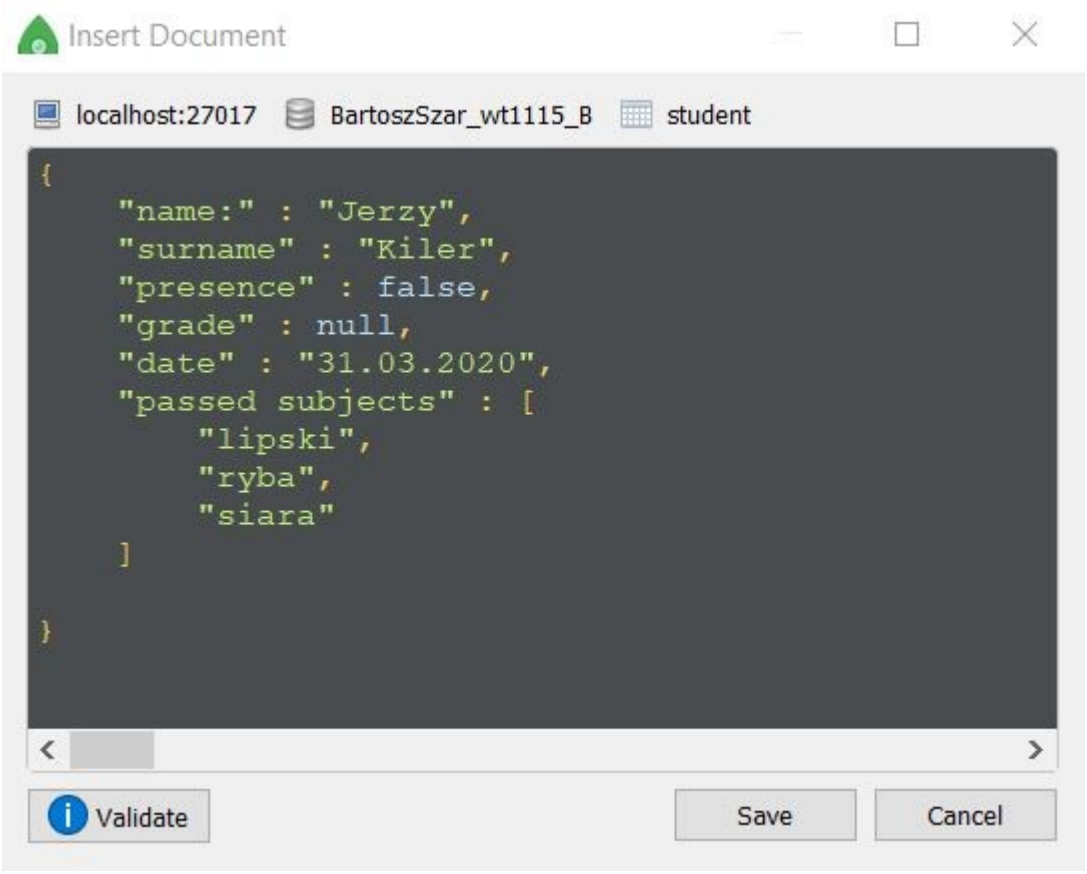
```
> show dbs
BartoszSzar_wt1115_B  0.653GB
admin                 0.000GB
config                0.000GB
local                 0.000GB
> use BartoszSzar_wt1115_B
switched to db BartoszSzar_wt1115_B
> show collections
business
checkin
review
tip
user
>
```

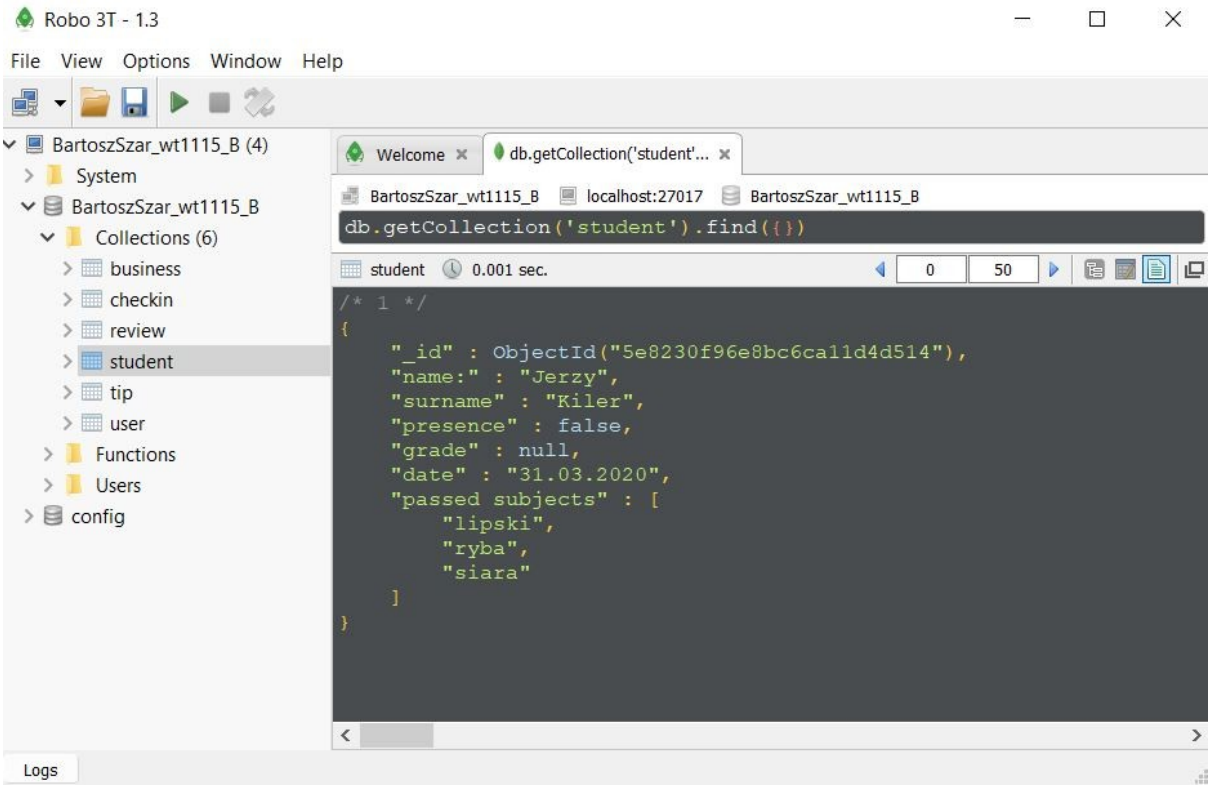
zad3

Po zainstalowaniu oprogramowania Robo3T podłączyłem do systemu utworzoną uprzednio w MongoDB bazę danych:

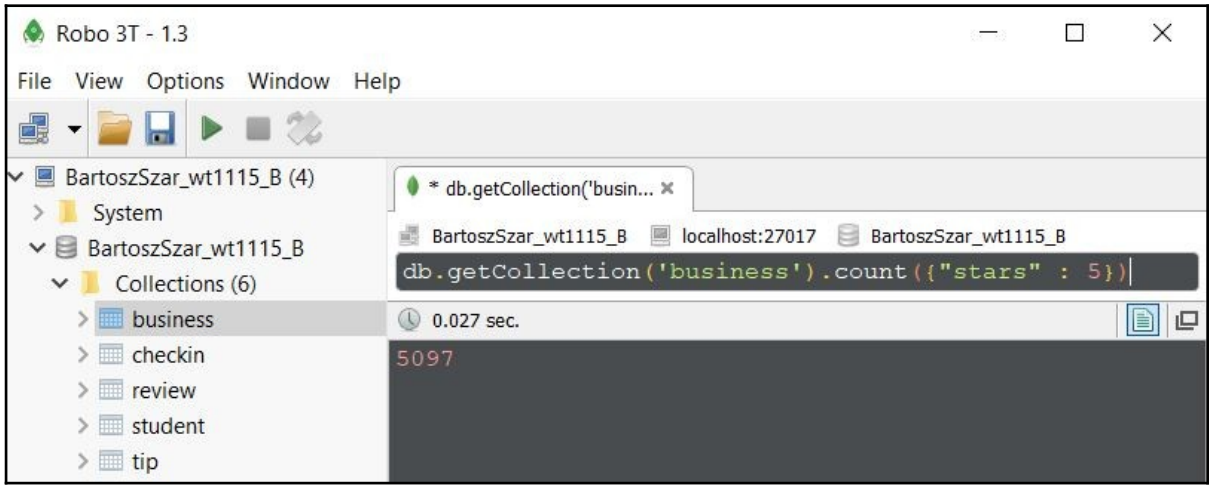


zad4

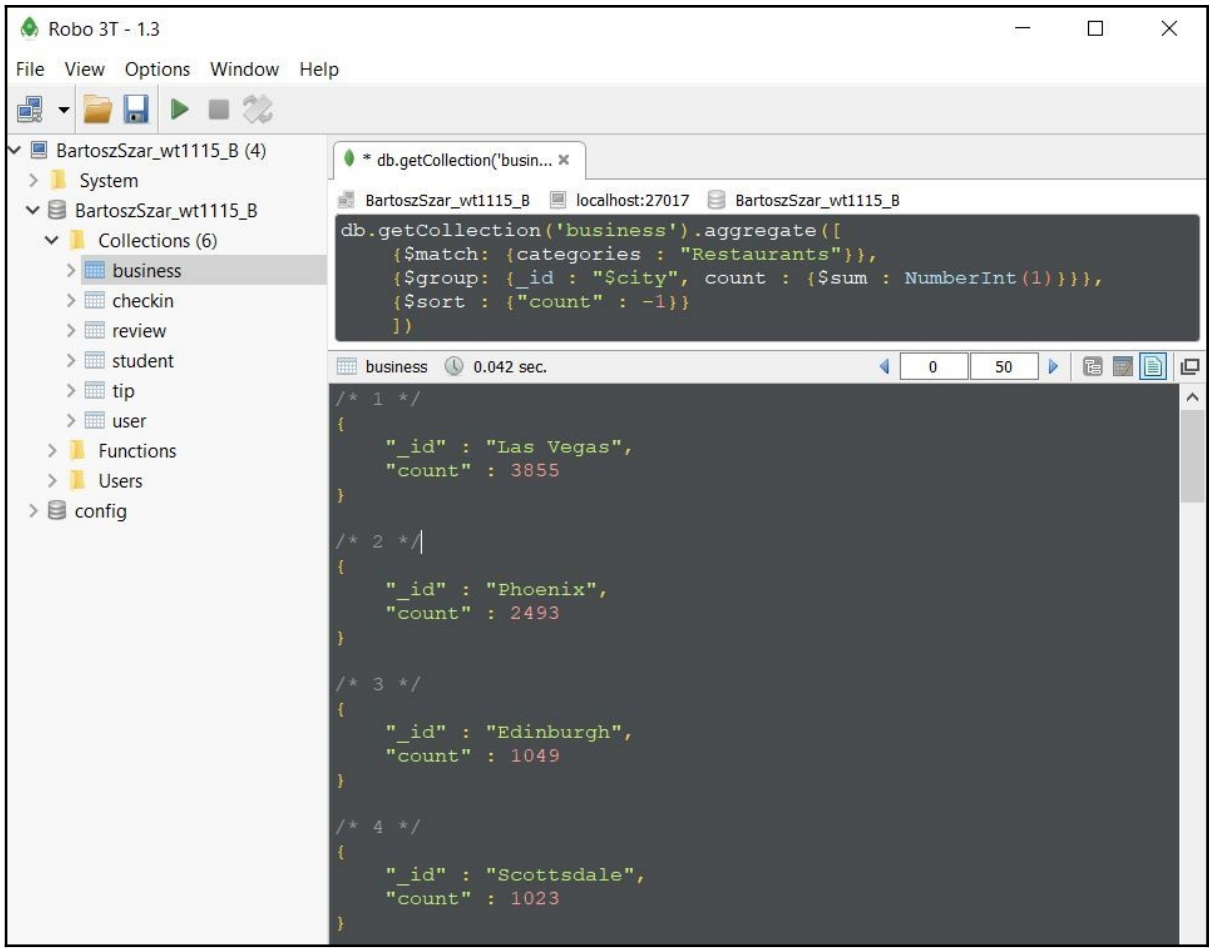

Insert Document
localhost:27017
BartoszSzar_wt1115_B
student
{
"name:" : "Jerzy",
"surname" : "Kiler",
"presence" : false,
"grade" : null,
"date" : "31.03.2020",
"passed subjects" : [
"lipski",
"ryba",
"siara"
]
}
Validate
Save
Cancel


Robo 3T - 1.3
File View Options Window Help
BartoszSzar_wt1115_B (4)
System
BartoszSzar_wt1115_B
Collections (6)
business
checkin
review
student
tip
user
Functions
Users
config
Welcome
db.getCollection('student'...
BartoszSzar_wt1115_B
localhost:27017
BartoszSzar_wt1115_B
db.getCollection('student').find({})
student
0.001 sec.
0
50
/* 1 */
{
"_id" : ObjectId("5e8230f96e8bc6ca11d4d514"),
"name:" : "Jerzy",
"surname" : "Kiler",
"presence" : false,
"grade" : null,
"date" : "31.03.2020",
"passed subjects" : [
"lipski",
"ryba",
"siara"
]
}
Logs

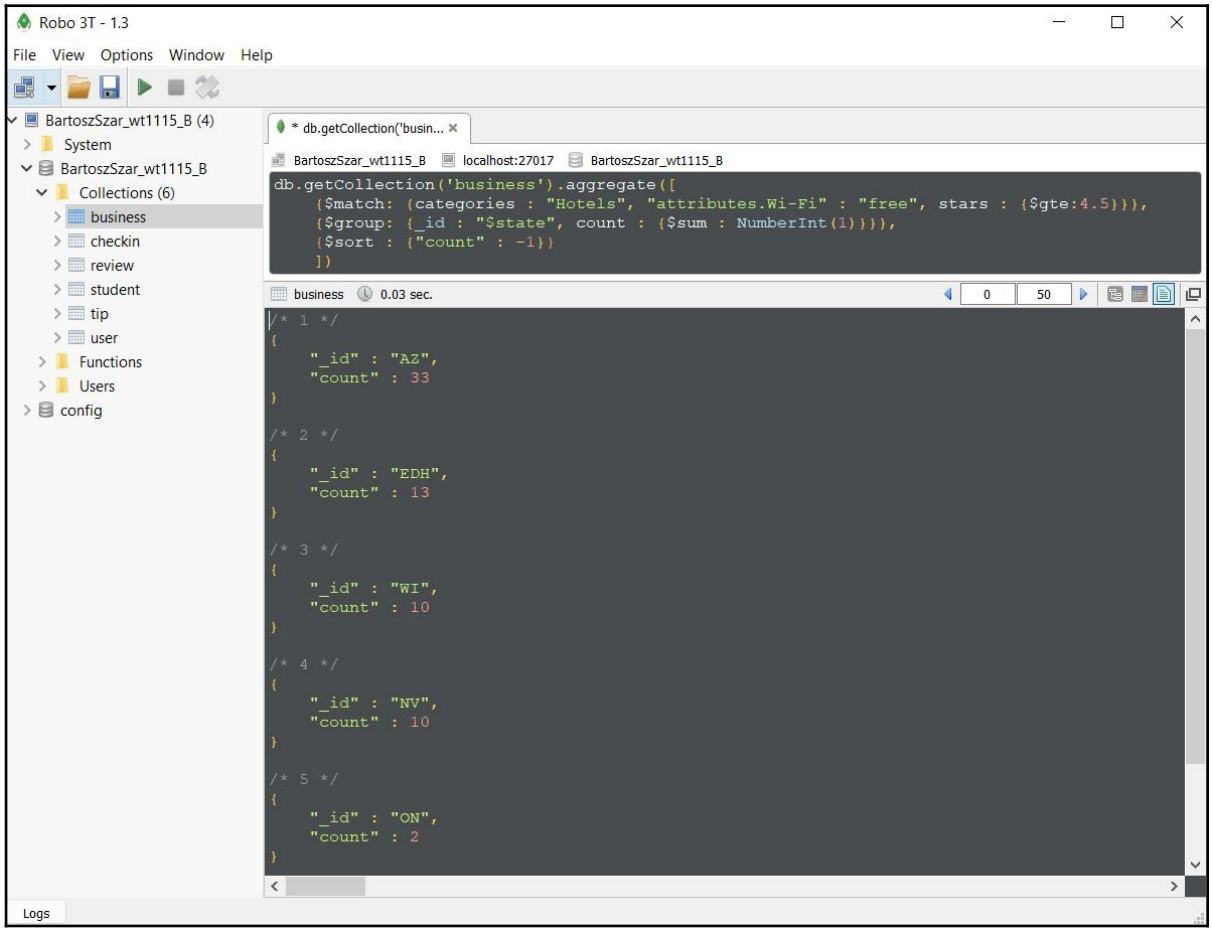
zad5 a)


Robo 3T - 1.3
File View Options Window Help
BartoszSzar_wt1115_B (4)
System
BartoszSzar_wt1115_B
Collections (6)
business
checkin
review
student
tip
* db.getCollection('busin...
BartoszSzar_wt1115_B
localhost:27017
BartoszSzar_wt1115_B
db.getCollection('business').count({"stars" : 5})
0.027 sec.
5097

zad5 b)


Robo 3T - 1.3
File View Options Window Help
BartoszSzar_wt1115_B (4)
System
BartoszSzar_wt1115_B
Collections (6)
business
checkin
review
student
tip
user
Functions
Users
config
* db.getCollection('busin...
BartoszSzar_wt1115_B
localhost:27017
BartoszSzar_wt1115_B
db.getCollection('business').aggregate([
{$match: {categories : "Restaurants"}},
{$group: {_id : "$city", count : {$sum : NumberInt(1)}}},
{$sort : {"count" : -1}}
])
business
0.042 sec.
0
50
/* 1 */
{
"_id" : "Las Vegas",
"count" : 3855
}
/* 2 */
{
"_id" : "Phoenix",
"count" : 2493
}
/* 3 */
{
"_id" : "Edinburgh",
"count" : 1049
}
/* 4 */
{
"_id" : "Scottsdale",
"count" : 1023
}

zad5 c)



zad6

```
protected long zad_6_5a(){
        MongoCollection<Document> business =
        database.getCollection("business");
        Document query = new Document("stars", 5);
        return business.count(query);
}
```

```
5097
```

```
protected void zad_6_5b(){
        MongoCollection<Document> business =
        database.getCollection("business");
        AggregateIterable<Document> result = business.aggregate(
                Arrays.asList(
                 Aggregates.match(Filters.eq("categories",
                        "Restaurants")),
                 Aggregates.group("$city",
                           Accumulators.sum("no_of_restaurants", 1)),
                  Aggregates.sort(Sorts.descending("no_of_restaurants"))
                )
        );
        for (Object tuple : result){
            System.out.println(tuple);
        }
}
```

```
Document{{_id=Las Vegas, no_of_restaurants=3855}}
Document{{_id=Phoenix, no_of_restaurants=2493}}
Document{{_id=Edinburgh, no_of_restaurants=1049}}
Document{{_id=Scottsdale, no_of_restaurants=1023}}
Document{{_id=Mesa, no_of_restaurants=693}}
Document{{_id=Madison, no_of_restaurants=679}}
Document{{_id=Tempe, no_of_restaurants=672}}
Document{{_id=Henderson, no_of_restaurants=564}}
Document{{_id=Chandler, no_of_restaurants=548}}
Document{{_id=Glendale, no_of_restaurants=422}}
Document{{_id=Gilbert, no_of_restaurants=317}}
Document{{_id=Peoria, no_of_restaurants=221}}
Document{{_id=North Las Vegas, no_of_restaurants=198}}
Document{{_id=Surprise, no_of_restaurants=144}}
Document{{_id=Goodyear, no_of_restaurants=119}}
Document{{_id=Waterloo, no_of_restaurants=117}}
Document{{_id=Avondale, no_of_restaurants=100}}
Document{{_id=Kitchener, no_of_restaurants=96}}
Document{{_id=Queen Creek, no_of_restaurants=82}}
Document{{_id=Middleton, no_of_restaurants=66}}
Document{{_id=Cave Creek, no_of_restaurants=63}}
Document{{_id=Casa Grande, no_of_restaurants=61}}
Document{{_id=Fountain Hills, no_of_restaurants=47}}
Document{{_id=Apache Junction, no_of_restaurants=44}}
Document{{_id=Buckeye, no_of_restaurants=42}}
```

```
protected void zad_6_5c(){
        MongoCollection<Document> business =
            database.getCollection("business");
        AggregateIterable<Document> result = business.aggregate(
                Arrays.asList(
                        Aggregates.match(Filters.and(
                                        Filters.eq("categories",
                                        "Hotels"),
                                        Filters.eq("attributes.Wi-Fi",
                                        "free"),
                                        Filters.gte("stars", 4.5))),
                        Aggregates.group("$state",
                                    Accumulators.sum("no_of_hotels", 1)),
                        Aggregates.sort(Sorts.descending("no_of_hotels"))
                )
        );
        for (Object tuple : result){
            System.out.println(tuple);
        }
}
```

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

zad 7

```
protected String zad7(){
        MongoCollection<Document> reviews =
            database.getCollection(("review"));
        MongoCollection<Document> users =
            database.getCollection(("user"));
        Object bestRatedUser = reviews.aggregate(
                Arrays.asList(
                      Aggregates.match(Filters.gte("stars", 4.5)),
                      Aggregates.group("$user_id",
                              Accumulators.sum("no_of_reviews", 1)),
                      Aggregates.sort(Sorts.descending("no_of_reviews"))
                )).first().get("_id");
        return users.find(Filters.eq("user_id",
        bestRatedUser.toString())).first().get("name").toString();
}
```

```
Rand
```

zad8

```
List<String> zad8(){
            MongoCollection<Document> reviews =
                database.getCollection(("review"));
            List<String> no_of_votes = new ArrayList<>();

            no_of_votes.add("funny: " + reviews.countDocuments(
                    Filters.gt("votes.funny", 0)
            ));
            no_of_votes.add("cool: " + reviews.countDocuments(
                    Filters.gt("votes.cool", 0)
            ));
            no_of_votes.add("useful: " + reviews.countDocuments(
                    Filters.gt("votes.useful", 0)
            ));

            return no_of_votes;
    }
```

```
[funny: 269256, cool: 346519, useful: 549519]
```