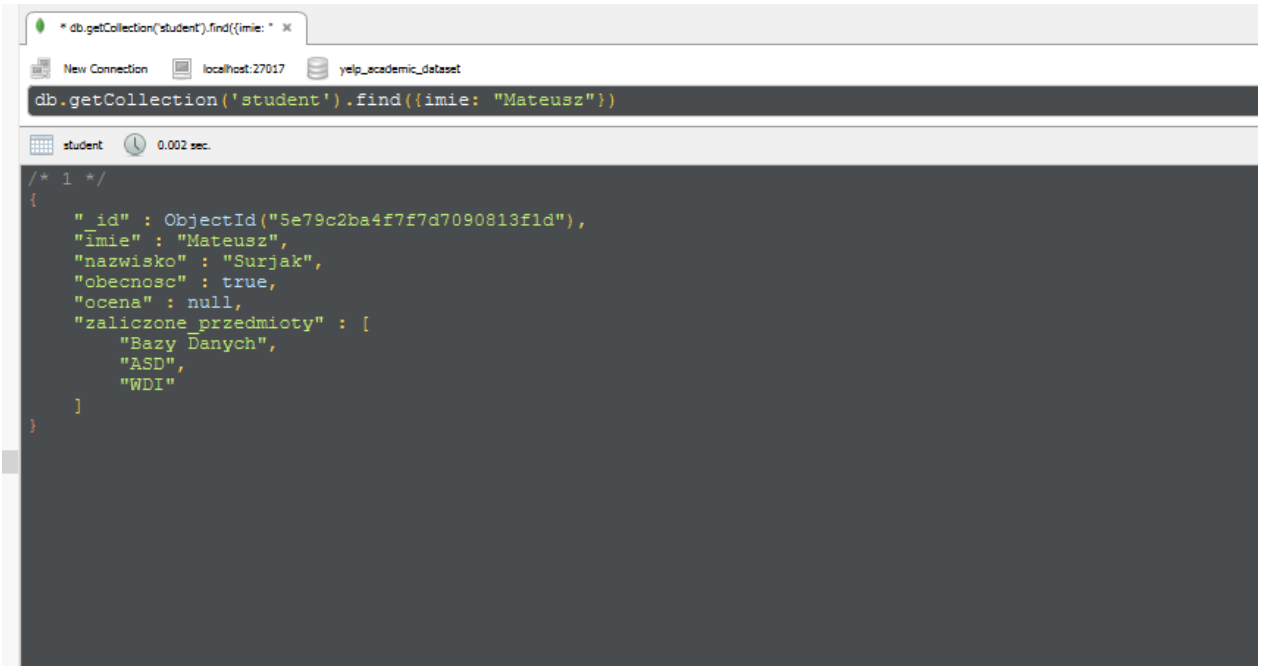
Laboratorium 2: MongoDB

Mateusz Surjak

1 zad4

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.

```
db.students.insert({
      "imie" : "Mateusz",
    "nazwisko" : "Surjak",
    "obecnosc" : true,
    "ocena" : null,
    "zaliczone_przedmioty" : [
        "Bazy Danych",
        "ASD",
        "WDI"
    ]
})
```



2 zad5

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

2.1 a

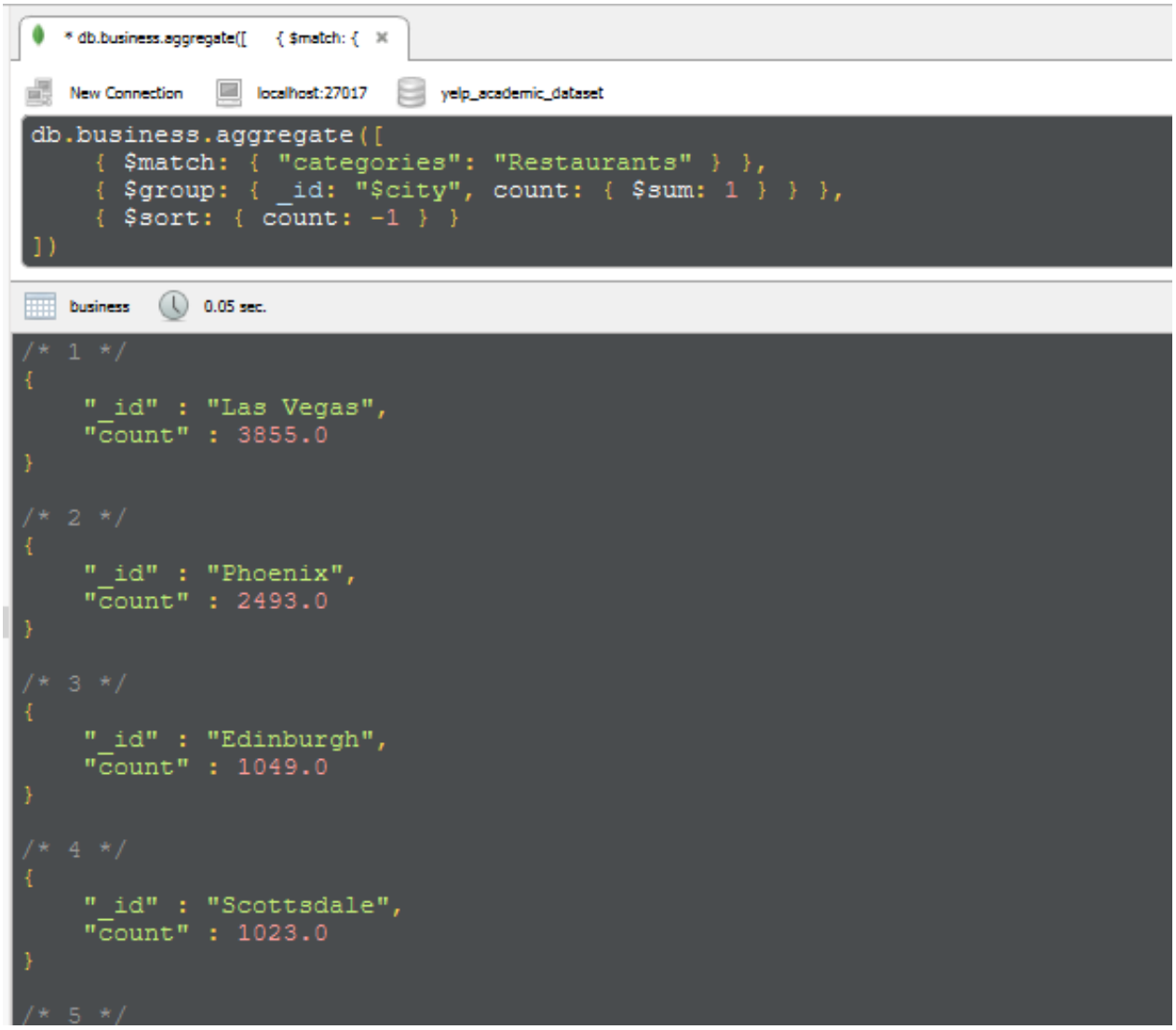
a. ilość miejsc ocenianych na 5 gwiazdek (pole stars, kolekcja business)

```
db.getCollection('business').find({stars: 5.0}).count()
```

2.2 b

b. ilość restauracji w każdym mieście, wynik posortuj malejąco na podstawie liczby. Pole categories w dokumencie business musi zawierać wartość Restaurants. Wykorzystaj operator group i funkcje aggregate.

```
db.business.aggregate([
    { $match: { "categories": "Restaurants" } },
    { $group: { _id: "$city", count: { $sum: 1 } } },
    { $sort: { count: -1 } }
])
```



2.3 c

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 operator group i funkcje aggregate.

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

])
```

New Connection | localhost:27017 | yelp_academic_dataset

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

])
```

business | 0.04 sec.

```
/* 1 */
{
    "_id" : "MLN",
    "count" : 1.0
}

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

/* 3 */
{
    "_id" : "ON",
    "count" : 2.0
}

/* 4 */
{
    "_id" : "NV",
    "count" : 10.0
}

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

/* 6 */
{
    "_id" : "AZ",
    "count" : 33.0
}
```

3 zad6

Wykonaj zadania punktu 5 z poziomu języka Java:

3.1 0

```java
public class MongoLab {
    private MongoClient mongoClient;
    private MongoDatabase db;

    public MongoLab() throws UnknownHostException {
        mongoClient = new MongoClient();
        db = mongoClient.getDatabase( databaseName: "yelp_academic_dataset");
    }

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

```
colname: checkin
colname: tip
colname: student
colname: business
colname: user
colname: review
```

3.2 a

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

```java
    long getCountOf5StarsBusiness(){
        MongoCollection<Document> collection = db.getCollection( s: "business");
        BasicDBObject whereStatement = new BasicDBObject();
        whereStatement.put("stars", 5.0);
        return collection.countDocuments(whereStatement);
    }
```

```
5097


Process finished with exit code 0
```

3.3 b

ilość restauracji w każdym mieście, wynik posortuj malejąco na podstawie liczby. Pole categories w dokumencie business musi zawierać wartość Restaurants. Wykorzystaj operator group i funkcje aggregate.

```
void getRestaurantsInCities(){
    MongoCollection<Document> collection = db.getCollection( s: "business");
    List agregateList = 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 iter = collection.aggregate(agregateList);

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

```
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}}
inal  0: Messages  4: Run  6: TODO
```

3.4 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 operator group i funkcje aggregate.

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

    );
    AggregateIterable iter = collection.aggregate(agregateList);
    for(Object i: iter){
        System.out.println(i);
    }
}
```


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

    );
    AggregateIterable iter = collection.aggregate(agregateList);
    for(Object i: iter){
        System.out.println(i);
    }
}
```

4 zad7

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

```
String getPositiveUser(){
    MongoCollection<Document> reviews = db.getCollection( s: "review");
    Object userID = reviews.aggregate(Arrays.asList(
            Aggregates.match(Filters.gte( fieldName: "stars",  value: 4.5)),
            Aggregates.group( id: "$user_id", Accumulators.sum( fieldName: "count",  expression: 1)),
            Aggregates.sort(Sorts.descending( ...fieldNames: "count"))
    )).first().get("_id");

    return db.getCollection( s: "user").find(Filters.eq( fieldName: "user_id", userID.toString())).first().get("name").toString();

}
```

```
Rand


Process finished with exit code 0
```

5 zad8

Napisz kod w języku Java, który 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).

```
StringBuilder countVotesReviews(){
    MongoCollection<Document> reviews = db.getCollection( s: "review");
    StringBuilder sb = new StringBuilder();
    long funny = reviews.countDocuments(
            Filters.gt( fieldName: "votes.funny",  value: 0)
    );
    long cool = reviews.countDocuments(
            Filters.gt( fieldName: "votes.cool",  value: 0)
    );
    long useful = reviews.countDocuments(
            Filters.gt( fieldName: "votes.useful",  value: 0)
    );
    sb.append("useful: " + useful+ "\n");
    sb.append("funny: " + funny+ "\n");
    sb.append("cool: " + cool+ "\n");
    System.out.println(sb);
    return sb;
}
```

```
useful: 549519
funny: 269256
cool: 346519


Process finished with exit code 0
```