

Laboratorium 2: MongoDB

Mateusz Surjak

Tydzień A Wt 9:35

1 Zadanie 1

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

1.1 a

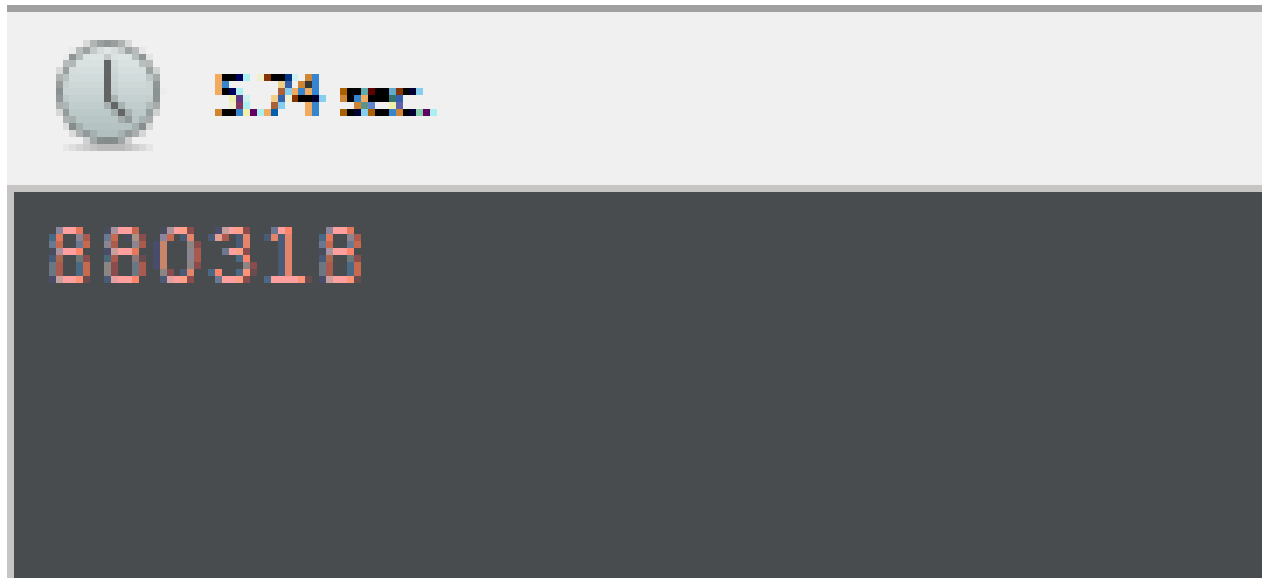
Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
1 db.business.distinct("city").sort()
```

```
[
  "Ahwatukee",
  "Anthem",
  "Apache Junction",
  "Arcadia",
  "Atlanta",
  "Avondale",
  "Black Canyon City",
  "Bonnyrigg",
  "Boulder City",
  "Buckeye",
  "C Las Vegas",
  "Cambridge",
```

1.2 b

```
1 db.review.find({"date": {$gte: '2011-01-01'}}).count()
```



1.3 c

Zwróć dane wszystkich zamkniętych (open) firm (business) z pól: nazwa, adres, gwiazdki (stars).

```
1 db.business.find({open: false}, {name: 1, stars:1, full_address:1})
```

```
/* 1 */
{
  "_id" : ObjectId("5e79177dc0e935d4a883dc90"),
  "full_address" : "6401 University Ave\nMiddleton, WI 53562",
  "name" : "Crandalls Carryout & Catering",
  "stars" : 4.0
}

/* 2 */
{
  "_id" : ObjectId("5e79177dc0e935d4a883dc98"),
  "full_address" : "6230 University Ave\nMiddleton, WI 53562",
  "name" : "Mi Cocina",
  "stars" : 3.0
}

/* 3 */
{
  "_id" : ObjectId("5e79177dc0e935d4a883dc9a"),
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
  "name" : "Charter Communications",
}
```

1.4 d

Zwróć dane wszystkich użytkowników (user), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (funny lub useful), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

```
1 db.user.find({ $or: [{"votes.funny": 0}, {"votes.useful": 0}]}).sort({"name": 1})
```

```
/* 1 */
{
  "_id" : ObjectId("5e791725c0e935d4a8804444"),
  "yelping_since" : "2009-08",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "Bernard",
  "user_id" : "xP3SPgfgW2vc5Zj5uV8SEA",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}

/* 2 */
{
  "_id" : ObjectId("5e791728c0e935d4a88142ac"),
  "yelping_since" : "2011-12",
  "votes" : {
    "funny" : 0,
    "useful" : 2,
    "cool" : 2
  },
  "review_count" : 10,
  "name" : "Anastacia",
  "user_id" : "qJLc0rYytqzeVBtTPFtfSA",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 4.5,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}

/* 3 */
{
  "_id" : ObjectId("5e791724c0e935d4a880104d"),
  "yelping_since" : "2012-03",
  "votes" : {
    "funny" : 0,
    "useful" : 1,
    "cool" : 1
  },
  "review_count" : 2,
  "name" : "Brandon",
  "user_id" : "CfWeBCoVvHze8bK WSDcEq",
}
```

1.5 e

Określ, ile każde przedsiębiorstwo otrzymało wskazówek napiwków (tip) w 2012. Wynik posortuj alfabetycznie na podstawie liczby (tip).

```
1 db.tip.aggregate([
2   {$match:{"date":"/2012/"}}},
3   {$group : {_id:"$business_id", count:{$sum:1}}},
4   {$sort:{"count":1}}
5 ])
```

```
/* 1 */
{
  "_id" : "yCaSjO3AHkirEyQ074cp4Q",
  "count" : 1.0
}

/* 2 */
{
  "_id" : "gLvUx5L7awAUIxK7rp6NA",
  "count" : 1.0
}

/* 3 */
{
  "_id" : "CbUkl9BpdI_m7Yt4yTca_Q",
  "count" : 1.0
}

/* 4 */
{
  "_id" : "ToSm1CIH3xgaedt63f3FCQ",
  "count" : 1.0
}

/* 5 */
{
  "_id" : "4KWcSe7p5qUIabx7f0Yu9w",
  "count" : 1.0
}

/* 6 */
{
  "_id" : "16XWVH1K-X6E-374-XD1-4"
```

1.6 f

Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji. Wynik ogranicz do recenzji, które uzyskały min 4.0 gwiazdki.

```
1 db.review.aggregate([
2   $group:
3   {
4     _id: "$business_id",
5     avgStars: {$avg: "$stars"}
6   },
7 ],
8 { $match:
9   { avgStars: { $gte: 4 }
10 }
11 }
12 ])
```

```
/* 1 */
{
  "_id" : "nYer89hXYAoddMEKTxw7kA",
  "avgStars" : 4.916666666666667
}

/* 2 */
{
  "_id" : "tsvWY4o64xiv7K0VA89R8A",
  "avgStars" : 5.0
}

/* 3 */
{
  "_id" : "uUsfpN81JCMKyH6c0D0bTg",
  "avgStars" : 4.0
}

/* 4 */
{
  "_id" : "iyKyJoDcbkGrMCgvyFMHxw",
  "avgStars" : 5.0
}

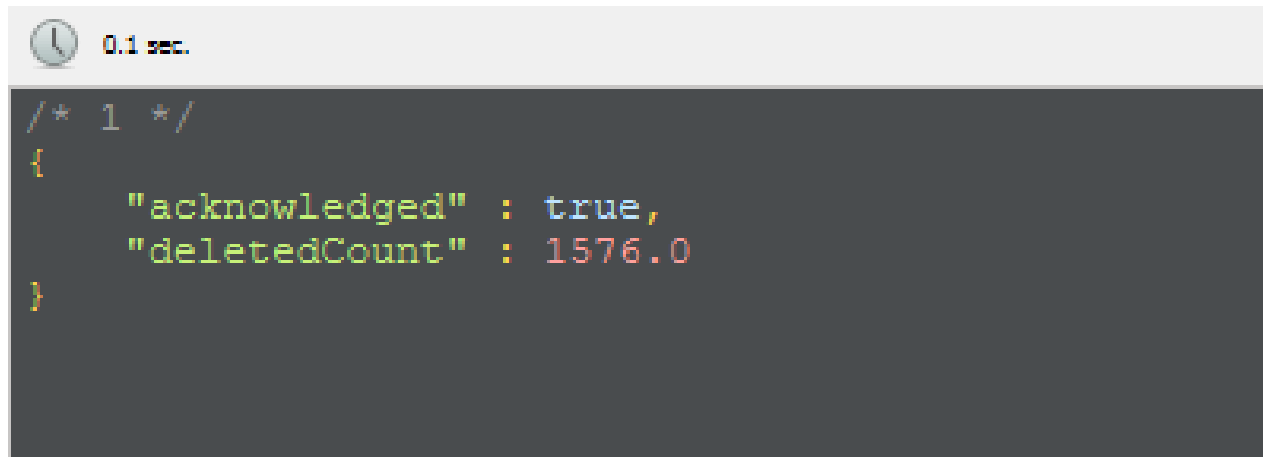
/* 5 */
{
  "_id" : "UH4BReVAqpMVjyVroEJAGA",
  "avgStars" : 4.0
}

/* 6 */
{
  "_id" : "flhtNfzfGiySc3g_U0Zkeg",
  "avgStars" : 5.0
}
```

1.7 g

Usuń wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0.

```
1 db.business.deleteMany({stars: 2.0})
```

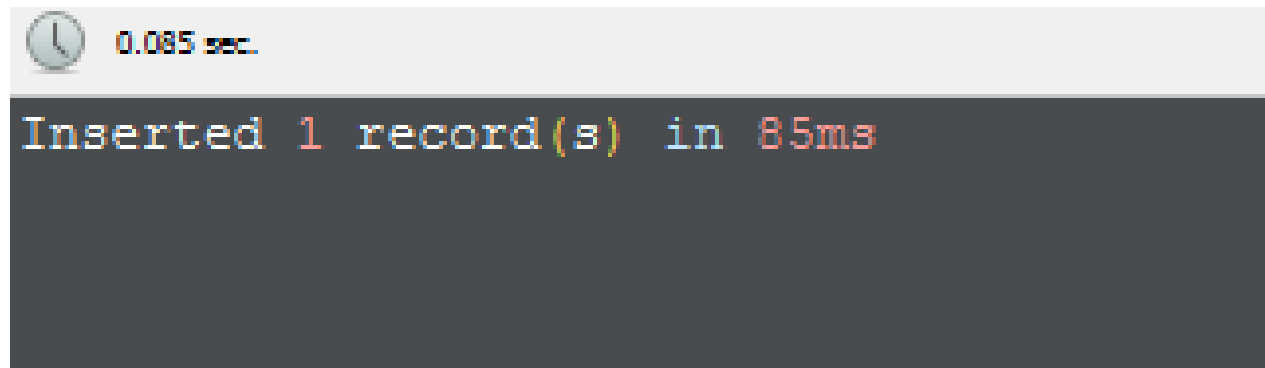


A screenshot of the MongoDB Shell interface. At the top, a status bar shows a clock icon and the text "0.1 sec.". Below this, the command `/* 1 */` is entered. The result is a JSON object: `{ "acknowledged" : true, "deletedCount" : 1576.0 }`. The text is color-coded: `/*` is blue, `1` is green, `*/` is blue, `{` is orange, `"acknowledged"` is green, `:` is blue, `true` is blue, `,` is blue, `"deletedCount"` is green, `:` is blue, and `1576.0` is red. The closing brace `}` is orange.

2 Zadanie 2

Zdefiniuj funkcję (MongoDB) umożliwiającą dodanie nowej recenzji (review). Wykonaj przykładowe wywołanie.

```
1 function addReview(user_id, review_id, text, business_id){
2     db.review.insert({
3         votes: {
4             funny: 0,
5             useful: 0,
6             cool: 0
7         },
8         user_id: user_id,
9         review_id: review_id,
10        stars : 0,
11        date : new Date(),
12        text : text,
13        type: "review",
14        business_id :business_id
15    })
16 }
17
18 addReview("zvJCcrpm2yOZrxKffwGQLA", "-TsVN230RCkLYKBeLsuz7A", "Some
  ↳ text", "vcNAWiLM4dR7D2nwwJ7nCA" )
```



3 Zadanie 3

Zdefiniuj funkcję (MongoDB), która zwróci wszystkie biznesy (business), w których w kategorii znajduje się podana przez użytkownika cecha. Wartość kategorii należy przekazać do funkcji jako parametr. Wykonaj przykładowe wywołanie zdefiniowanej funkcji.

```
1 function getBusinessWithCategory(category){
2     return db.business.find({
3         categories: category
4     })
5
6 }
7
8 getBusinessWithCategory("Doctors")
```

```
/* 1 */
{
  "_id" : ObjectId("5e79177dc0e935d4a883dc88"),
  "business_id" : "vcNAW1LM4dR7D2nwwJ7nCA",
  "full_address" : "4840 E Indian School Rd\nSte 101\nPhoenix, AZ 85018",
  "hours" : {
    "Tuesday" : {
      "close" : "17:00",
      "open" : "08:00"
    },
    "Friday" : {
      "close" : "17:00",
      "open" : "08:00"
    },
    "Monday" : {
      "close" : "17:00",
      "open" : "08:00"
    },
    "Wednesday" : {
      "close" : "17:00",
      "open" : "08:00"
    },
    "Thursday" : {
      "close" : "17:00",
      "open" : "08:00"
    }
  },
  "open" : true,
  "categories" : [
    "Doctors",
    "Health & Medical"
  ],
  "city" : "Phoenix",
  "review_count" : 7,
  "name" : "Eric Goldberg, MD",
  "neighborhoods" : [],
  "longitude" : -111.983758,
  "state" : "AZ",
  "stars" : 3.5,
  "latitude" : 33.499313,
  "attributes" : {
    "By Appointment Only" : true
  },
  "type" : "business"
}

/* 2 */
{
  "_id" : ObjectId("5e79177dc0e935d4a883ddf9"),
  "business_id" : "PFO7RF1xz4ZBuag46KsLpA",
  "full_address" : "2332 N Central Ave\nPhoenix, AZ 85004",
```

4 Zadanie 4

Zdefiniuj funkcję (MongoDB), która umożliwi modyfikację nazwy użytkownika (user) na podstawie podanego id. Id oraz nazwa mają być przekazywane jako parametry.

```
1 function modifyUser(id, newName){
2     db.user.findOneAndUpdate({_id: new ObjectId(id)}, {$set: {name: newName}})
3
4 }
5
6 modifyUser("5e791722c0e935d4a87f83f9", "Mateusz")
```

```
function modifyUser(id, newName){
    db.user.findOneAndUpdate({_id: new ObjectId(id)}, {$set: {name: newName}})
}
|
modifyUser("5e791722c0e935d4a87f83f9", "Mateusz")
db.user.find({_id: new ObjectId("5e791722c0e935d4a87f83f9")})
```

user 0.001 sec.

```
/* 1 */
{
  "_id" : ObjectId("5e791722c0e935d4a87f83f9"),
  "yelping_since" : "2012-02",
  "votes" : {
    "funny" : 1,
    "useful" : 5,
    "cool" : 0
  },
  "review_count" : 6,
  "name" : "Mateusz",
  "user_id" : "qtrmBGNqCvupHMHL_bKFgQ",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 3.83,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}
```

5 Zadanie 5

Zwróć średnią ilość wszystkich wskazówek/napiwków dla każdego z biznesów, wykorzystaj map reduce.

```
1  var mapF = function(){
2  var key = this.business_id;
3  var value = {
4      count: 1
5  }
6      emit(key,value);
7  };
8
9  var reduceF = function(key, values){
10     var counter = 0;
11     values.forEach(function(val){counter+=val.count;})
12     return counter;
13
14 }
15
16 var finalizeFunction = function(key, reducedValue)
17 {
18
19     return reducedValue;
20 }
21
22
23 db.tip.mapReduce(
24 mapF,reduceF, {out: "count", finalize:finalizeFunction }
25
26 )
27 db.count.find()
```

```

/* 1 */
{
  "result" : "count",
  "timeMillis" : 4779.0,
  "counts" : {
    "input" : 403210,
    "emit" : 403210,
    "reduce" : 23629,
    "output" : 29909
  },
  "ok" : 1.0,
  "_o" : {
    "result" : "count",
    "timeMillis" : 4779,
    "counts" : {
      "input" : 403210,
      "emit" : 403210,
      "reduce" : 23629,
      "output" : 29909
    },
    "ok" : 1.0
  },
  "_keys" : [
    "result",
    "timeMillis",
    "counts",
    "ok"
  ],
  "_db" : {
    "_mongo" : {
      "slaveOk" : true,
      "host" : "localhost:27017",
      "name" : "test"
    }
  }
}

```

```

/* 1 */
{
  "_id" : "--1emggGHgoG6ipd_RMb-g",
  "value" : 6.0
}

/* 2 */
{
  "_id" : "--5jkZ3-nUPZxUvtcbr8Uw",
  "value" : 16.0
}

/* 3 */
{
  "_id" : "--BlvDO_RG2yElKu9XA1_g",
  "value" : 21.0
}

/* 4 */
{
  "_id" : "--Dl2rW_xO8GuYBomlg9zw",
  "value" : 2.0
}

/* 5 */
{
  "_id" : "--Y_2lD0tVDioX5bwF6GIw",
  "value" : 5.0
}

/* 6 */
{

```

6 Zadanie 6

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

6.1 a

Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (business). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
1 public void getCities() {
2     ArrayList cities = db.getCollection("business")
3         .distinct("city", String.class)
4         .into(new ArrayList<>());
5
6     Collections.sort(cities);
7
8     cities.forEach(System.out::println);
9 }
```

```
public void getCities(){
    ArrayList cities = db.getCollection( collectionName: "business")
        .distinct( fieldName: "city", String.class)
        .into(new ArrayList<>());

    Collections.sort(cities);

    cities.forEach(System.out::println);
}
```

```
Ahwatukee
Anthem
Apache Junction
Arcadia
Atlanta
Avondale
Black Canyon City
Bonnyrigg
Boulder City
Buckeye
```

6.2 b

Zwróć liczbę wszystkich recenzji, które pojawiły się po 2011 roku (włącznie).

```
1 public void getReviewsAfter2011() {
2     Document query = new Document("date", new Document("$gte", "2011-01-01"));
3     long count = db.getCollection("review").countDocuments(query);
4     System.out.println(count);
5 }
```

```
public void getReviewsAfter2011() {
    Document query = new Document("date", new Document("$gte", "2011-01-01"));
    long count = db.getCollection("review").countDocuments(query);
    System.out.println(count);
}
```

880318

6.3 c

Zwróć dane wszystkich zamkniętych (open) firm (business) z pól: nazwa, adres, gwiazdki (stars).

```
1 public void getClosedCompanies() {
2     db.getCollection("business")
3         .find(eq("open", false))
4         .projection(fields(include("name", "stars", "fill_address"),
5                             ↪ excludeId()))
6         .into(new ArrayList<>()).forEach(System.out::println);
7 }
```

```
public void getClosedCompanies() {
    db.getCollection(collectionName: "business") MongoClient<...>
        .find(eq(fieldName: "open", value: false)) FindIterable<...>
        .projection(fields(include(...fieldNames: "name", "stars", "fill_address"), excludeId())) FindIterable<...>
        .into(new ArrayList<>()).forEach(System.out::println);
}
```

```

Document{{name=Mom & Pop's Bagel & Bakery, stars=4.0}}
Document{{name=Jimmy G's Burgers, stars=4.0}}
Document{{name=Meatball Spot, stars=2.5}}
Document{{name=Dollar Bee, stars=3.5}}
Document{{name=F?n Boy at ASU, stars=4.5}}
Document{{name=S2pizzabar, stars=4.0}}
Document{{name=The Act Nightclub, stars=4.0}}
Document{{name=Pizza Buddha, stars=4.5}}
Document{{name=O Bar & Grill, stars=1.5}}
Document{{name=Le Passepartout, stars=4.0}}
Document{{name=My Big Fat Greek Restaurant, stars=4.0}}
Document{{name=Rattlecan, stars=4.0}}
Document{{name=Perfectly Polished Nails, stars=5.0}}
Document{{name=Korea House BBQ, stars=3.0}}

```

6.4 d

Zwróć dane wszystkich użytkowników (user), którzy nie uzyskali ani jednego pozytywnego głosu z kategorii (funny lub useful), wynik posortuj alfabetycznie na podstawie imienia użytkownika.

```

1 void getUsers() {
2
3     db.getCollection("user")
4         .find(or(eq("votes.funny", 0), eq("votes.useful", 0)))
5         .sort(ascending("name"))
6         .into(new ArrayList<>()).forEach(System.out::println);
7 }

```

```

void getUsers() {
    db.getCollection( collectionName: "user") MongoCollection<Document>
        .find(or( ...filters: eq( fieldName: "votes.funny", value: 0), eq( fieldName: "votes.useful", value: 0))) FindIterable<Document>
        .sort(ascending( ...fieldNames: "name")) FindIterable<Document>
        .into(new ArrayList<>()).forEach(System.out::println);
}

```

```

average_stars=3.2, type=user, compliments=Document{{note=1}}, elite=[]}}
Document{{_id=5e791724c0e935d4a8801560, yelping_since=2010-06, votes=Document{{funny=0, useful=1, cool=0}}, review_count=1, name=your-mom, user_id=6
average_stars=4.0, type=user, compliments=Document{{}}, elite=[]}}
Document{{_id=5e79172dc0e935d4a882f9d6, yelping_since=2011-03, votes=Document{{funny=0, useful=5, cool=0}}, review_count=13, name=yoyi, user_id=W2s5
average_stars=4.62, type=user, compliments=Document{{}}, elite=[]}}
Document{{_id=5e791722c0e935d4a87f8964, yelping_since=2010-10, votes=Document{{funny=0, useful=0, cool=0}}, review_count=3, name=yummy, user_id=KyA9
average_stars=3.0, type=user, compliments=Document{{}}, elite=[]}}
Document{{_id=5e79172ec0e935d4a883265f, yelping_since=2011-08, votes=Document{{funny=0, useful=7, cool=1}}, review_count=22, name=yvonneandscott, us
average_stars=3.45, type=user, compliments=Document{{}}, elite=[]}}
Document{{_id=5e79172bc0e935d4a8823a85, yelping_since=2010-03, votes=Document{{funny=0, useful=6, cool=0}}, review_count=5, name=zeus, user_id=8vRY9
average_stars=2.4, type=user, compliments=Document{{cool=1}}, elite=[]}}
Document{{_id=5e791729c0e935d4a8818559, yelping_since=2011-10, votes=Document{{funny=0, useful=2, cool=0}}, review_count=2, name=zhanna, user_id=mXh
average_stars=1.0, type=user, compliments=Document{{}}, elite=[]}}

```

6.5 e

Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (tip) w 2012. Wynik posortuj alfabetycznie na podstawie liczby (tip).

```
1 public void getTips(){
2     db.getCollection("tip").aggregate(Arrays.asList(
3         Aggregates.group("$business_id", Accumulators.sum("count",1))
4     )).into(new ArrayList<>()).forEach(System.out::println);
5 }
```

```
public void getTips(){
    db.getCollection( collectionName: "tip").aggregate(Arrays.asList(
        Aggregates.group( id: "$business_id", Accumulators.sum( fieldName: "count", expression: 1))
    )).into(new ArrayList<>()).forEach(System.out::println);
}
```

```
Document{{_id=nyjUGrVp0QgWepIX2Sang, count=2}}
Document{{_id=mHbC12IspsgkbhIoX1iD7w, count=1}}
Document{{_id=HS9zL2VJquqYi0UiIVvPqw, count=5}}
Document{{_id=PCu81PDqTlad-pbz988TKA, count=15}}
Document{{_id=jZ58NvLSZ6oVY1Rta1zUNQ, count=12}}
Document{{_id=MKsb2VpLB-0UBODcInDsSw, count=2}}
Document{{_id=DlCtdbceo4YNSI53cCL2lg, count=55}}
Document{{_id=TPmAm16wQ0tAFRq2uWHUHg, count=1}}
Document{{_id=A6DFzxQT5yBkwBBjsURRWQ, count=21}}
Document{{_id=BWTF6pUy7pptYNTfQVBiiQ, count=1}}
```

6.6 f

Wyznacz, jaką średnia ocen (stars) uzyskała każda firma (business) na podstawie wszystkich recenzji. Wynik ogranicz do recenzji, które uzyskały min 4.0 gwiazdki.

```
1 public void getAVGStars() {
2     db.getCollection("review").aggregate(Arrays.asList(
3         Aggregates.group("$business_id", Accumulators.avg("avgstars", "$stars")),
4         Aggregates.match(Filters.gte("avgstars", 4.0))
5     )).into(new ArrayList<>()).forEach(System.out::println);
6 }
```

```
public void getAVGStars() {
    db.getCollection( collectionName: "review").aggregate(Arrays.asList(
        Aggregates.group( id: "$business_id", Accumulators.avg( fieldName: "avgstars", expression: "$stars")),
        Aggregates.match(Filters.gte( fieldName: "avgstars", value: 4.0))
    )).into(new ArrayList<>()).forEach(System.out::println);
}
```






```
Document({_id=61C_xl0n3vKec0bT972x0g, avgstars=5.0}}
Document({_id=6Fu26Y978okFG6WgOMfu-Q, avgstars=4.884615384615385}}
Document({_id=M-vP3EVx__WpWspmj8IDcg, avgstars=5.0}}
Document({_id=wSZ6Aa1BR0488tZv1UuENw, avgstars=5.0}}
Document({_id=Z_NMfH1S3PbQMLvlqk_qLA, avgstars=4.5}}
Document({_id=T0g94VgTzo02XNm8MvTvZg, avgstars=4.923076923076923}}
Document({_id=X9savRKExqQsP9ZNSsz98g, avgstars=5.0}}
Document({_id=Zc4wit6xSj3fZZFICflsyw, avgstars=4.142857142857143}}
Document({_id=k7PlvJABB-sd2FK7A7ECtQ, avgstars=5.0}}
Document({_id=sTLaqlyAzaUBtyc9Qx8a-A, avgstars=4.5}}
Document({_id=YYm1mnmE36Ne7d5sT9huZQ, avgstars=4.269230769230769}}
Document({_id=qGffC4h6UOTSkLhzjj59FQ, avgstars=4.320158102766799}}
Document({_id=Dktk0WoXR7TZmWqj7B7cqQ, avgstars=4.833333333333333}}
Document({_id=PJTkNpQoINUFE9II1DSYLA, avgstars=4.777777777777778}}
```

6.7 g

Usuń wszystkie firmy (business), które posiadają ocenę (stars) równą 2.0.

```
1 public void deleteAllCompaniesWithNote2(){
2     db.getCollection("business").deleteMany(eq("stars", 2.0));
3 }
```

```
public void deleteAllCompaniesWithNote2(){
    db.getCollection( collectionName: "business").deleteMany(eq( fieldName: "stars", value: 2.0));
}
```

 New Connection  localhost:27017  yelp_academic_dataset

```
db.getCollection('business').find({stars: 2.0})
```

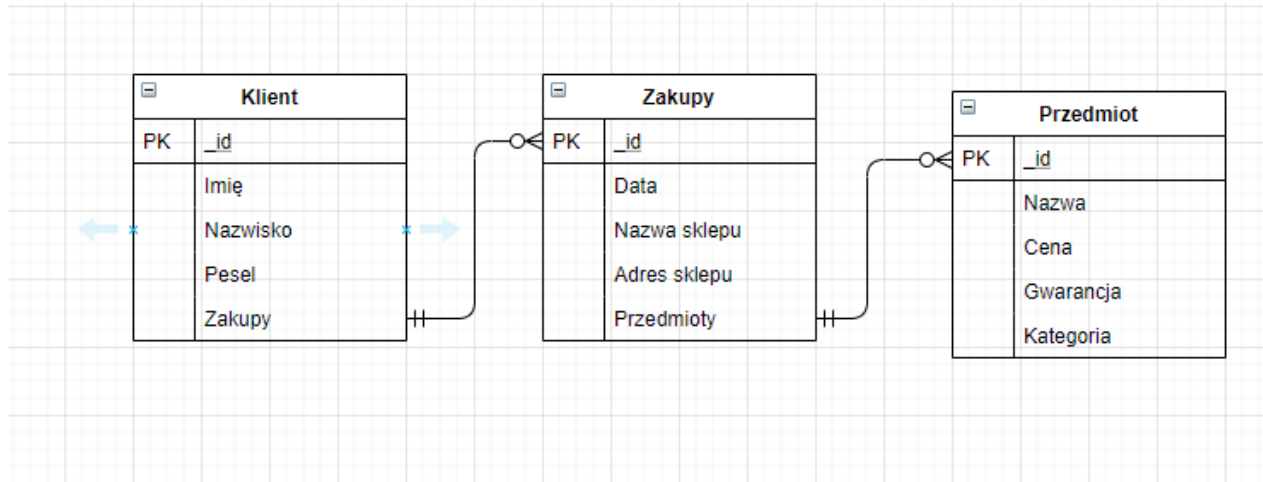
 0.021 sec.

Fetches 0 record(s) in 22ms

7 Zadanie 7

Zaproponuj bazę danych składającą się z 3 kolekcji pozwalającą przechowywać dane dotyczące: klientów, zakupu oraz przedmiotu zakupu. W bazie wykorzystaj: pola proste, złożone i tablice. Zaprezentuj strukturę dokumentów w formie JSON dla przykładowych danych. Uzasadnij swoją propozycję.

7.1 Schemat



7.2 Komentarz

Moją propozycją jest aby każdy klient miał listę zakupów które zrobił (jeden element z listy jest niejako paragonem sklepowym), oprócz tego klient posiada swoje dane osobowe. Każda pozycja odwołuje się do pojedynczych zakupów. Pojedyncze zakupy posiadają listę przedmiotów które klient zakupił w ramach tych zakupów. Każdy przedmiot na liście posiada cenę, nazwę oraz informację czy jest na gwarancji.

7.3 Klienci

```
1 function addClient(imie, nazwisko, pesel){
2   db.klienci.insert({
3     imie: imie,
4     nazwisko: nazwisko,
5     pesel: pesel,
6     zakupy: []
7   })
8 }
9 addClient("Adam", "Kowalski", 99032911846);
10 addClient("Karol", "Majewski", 91022911841);
11 addClient("Jan", "Nowak", 96012911841);
```

7.4 Zakupy

```
1 function addShopping(adresSklepu, nazwaSklepu){
2   db.zakupy.insert({
3     data: new Date(),
```

```

4     adresSklepu: adresSklepu,
5     nazwaSklepu: nazwaSklepu,
6     przedmioty: []
7   });
8 };
9
10 addShopping("Kosicicka 12", "Carrefour");
11 addShopping("Aleja Pokolu 67", "Auchan");
12 addShopping("Bonarka 12", "Real");

```

7.5 Przedmioty

```

1 function addProduct(nazwa, cena, gwarancja, kategoria){
2   db.przedmioty.insert({
3     nazwa: nazwa,
4     cena: cena,
5     gwarancja: gwarancja,
6     kategoria: kategoria
7   });
8 };
9
10 addProduct("Telewizor", 1200.99, true, "Elektronika");
11 addProduct("Spodnie", 59.99, false, "Odzież");
12 addProduct("Mleko", 3.99, false, "Żywność");

```

7.6 Dodaj przedmiot do zakupów

```

1 function addToCart(zakupyID, przedmiotID){
2
3   db.zakupy.update({_id: new ObjectId(zakupyID)},
4   {$addToSet: {
5     przedmioty: {$ref: "przedmioty", $id: new ObjectId(przedmiotID)}
6   }
7   });
8 };
9
10 addToCart("5e7e0acb984c083b58a8b3a3", "5e7e0ba8984c083b58a8b3a5");
11 addToCart("5e7e0acb984c083b58a8b3a3", "5e7e0ba8984c083b58a8b3a6");
12 addToCart("5e7e0acb984c083b58a8b3a4", "5e7e0ba8984c083b58a8b3a6");

```

7.7 Dodaj pzakupy do klienta

```

1 function addCartToClient(userID, zakupyID){
2
3   db.klienci.update({_id: new ObjectId(userID)},
4   {$addToSet: {
5     zakupy: {$ref: "zakupy", $id: new ObjectId(zakupyID)}
6   }
7   });
8 };
9
10

```

```
11 addCartToClient("5e7e08e8984c083b58a8b39f", "5e7e0acb984c083b58a8b3a3")
12
13 addCartToClient("5e7e090a984c083b58a8b3a0", "5e7e0acb984c083b58a8b3a4")
```

7.8 Dane w bazie danych

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

Klienci 0.001 sec.

```
/* 1 */
{
  "_id" : ObjectId("5e7e08e8984c083b58a8b39f"),
  "imie" : "Adam",
  "nazwisko" : "Kowalski",
  "pesel" : 99032911846.0,
  "zakupy" : [
    {
      "$ref" : "zakupy",
      "$id" : ObjectId("5e7e0acb984c083b58a8b3a3")
    }
  ]
}

/* 2 */
{
  "_id" : ObjectId("5e7e090a984c083b58a8b3a0"),
  "imie" : "Jan",
  "nazwisko" : "Nowak",
  "pesel" : 96012911841.0,
  "zakupy" : [
    {
      "$ref" : "zakupy",
      "$id" : ObjectId("5e7e0acb984c083b58a8b3a4")
    }
  ]
}

/* 3 */
{
  "_id" : ObjectId("5e7e091b984c083b58a8b3a1"),
  "imie" : "Karol",
  "nazwisko" : "Majewski",
  "pesel" : 91022911841.0,
  "zakupy" : []
}
```

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

```
zakupy 0.001 sec.

/* 1 */
{
  "_id" : ObjectId("5e7e0acb984c083b58a8b3a3"),
  "data" : ISODate("2020-03-27T14:16:43.637Z"),
  "adresSklepu" : "Aleja Pokolu 67",
  "nazwaSklepu" : "Auchan",
  "przedmioty" : [
    {
      "$ref" : "przedmioty",
      "$id" : ObjectId("5e7e0ba8984c083b58a8b3a5")
    },
    {
      "$ref" : "przedmioty",
      "$id" : ObjectId("5e7e0ba8984c083b58a8b3a6")
    }
  ]
}

/* 2 */
{
  "_id" : ObjectId("5e7e0acb984c083b58a8b3a4"),
  "data" : ISODate("2020-03-27T14:16:43.639Z"),
  "adresSklepu" : "Bonarka 12",
  "nazwaSklepu" : "Real",
  "przedmioty" : [
    {
      "$ref" : "przedmioty",
      "$id" : ObjectId("5e7e0ba8984c083b58a8b3a6")
    }
  ]
}

/* 3 */
{
  "_id" : ObjectId("5e7e0fae984c083b58a8b3a8"),
  "data" : ISODate("2020-03-27T14:37:34.603Z"),
  "adresSklepu" : "Kosicicka 12",
  "nazwaSklepu" : "Carrefour",
  "przedmioty" : []
}
```

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



przedmioty



0.001 sec.

```
/* 1 */
{
  "_id" : ObjectId("5e7e0ba8984c083b58a8b3a5"),
  "nazwa" : "Telewizor",
  "cena" : 1200.99,
  "gwarancja" : true,
  "kategoria" : "Elektronika"
}

/* 2 */
{
  "_id" : ObjectId("5e7e0ba8984c083b58a8b3a6"),
  "nazwa" : "Spodnie",
  "cena" : 59.99,
  "gwarancja" : false,
  "kategoria" : "Odzież"
}

/* 3 */
{
  "_id" : ObjectId("5e7e0ba8984c083b58a8b3a7"),
  "nazwa" : "Mleko",
  "cena" : 3.99,
  "gwarancja" : false,
  "kategoria" : "Żywność"
}
```