

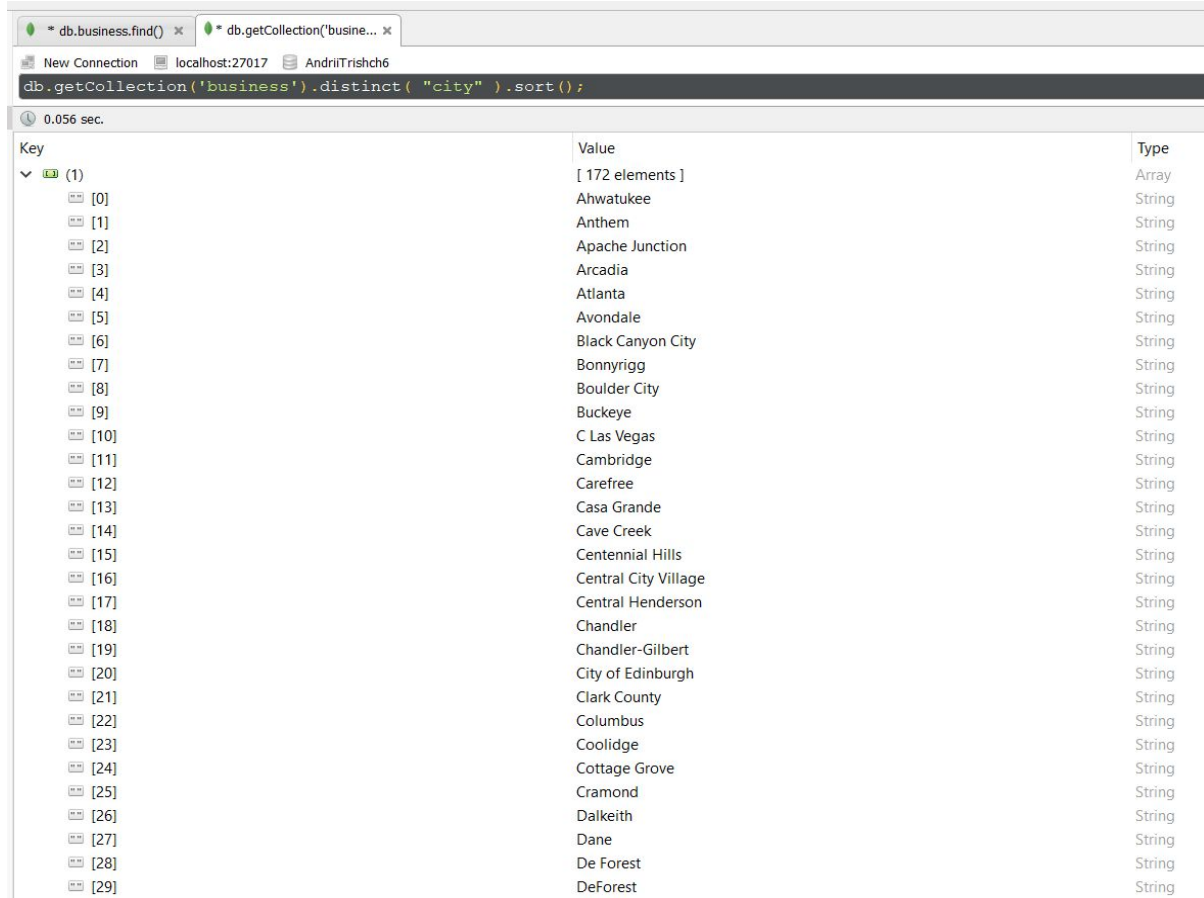
Bazy danych - NoSQL MongoDB - zadania

Imię i nazwisko: Andrii Trishch

Tydzień A/B, godz lab.: B/ Czw. 14:40

1. Wykorzystując bazę danych yelp dataset wykonaj zapytanie i komendy MongoDB, aby uzyskać następujące rezultaty
 - a) a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (*business*). Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
db.getCollection('business').distinct( "city" ).sort();
```



Key	Value	Type
(1)	[172 elements]	Array
[0]	Ahwatukee	String
[1]	Anthem	String
[2]	Apache Junction	String
[3]	Arcadia	String
[4]	Atlanta	String
[5]	Avondale	String
[6]	Black Canyon City	String
[7]	Bonnyrigg	String
[8]	Boulder City	String
[9]	Buckeye	String
[10]	C Las Vegas	String
[11]	Cambridge	String
[12]	Carefree	String
[13]	Casa Grande	String
[14]	Cave Creek	String
[15]	Centennial Hills	String
[16]	Central City Village	String
[17]	Central Henderson	String
[18]	Chandler	String
[19]	Chandler-Gilbert	String
[20]	City of Edinburgh	String
[21]	Clark County	String
[22]	Columbus	String
[23]	Coolidge	String
[24]	Cottage Grove	String
[25]	Cramond	String
[26]	Dalkeith	String
[27]	Dane	String
[28]	De Forest	String
[29]	DeForest	String

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

```
db.getCollection('review').  
  find({"date": { $gt: new Date('2011-01-01').toISOString()}}).count()
```

2.18 sec.

879931

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

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

Key	Value	Type
▼ (1) ObjectId("5ea186e8ad815bede96de947")	{ 4 fields }	Object
_id	ObjectId("5ea186e8ad815bede96de947")	ObjectId
full_address	4156 County Rd B Mc Farland, WI 53558	String
name	Charter Communications	String
stars	1.5	Double
▼ (2) ObjectId("5ea186e8ad815bede96de952")	{ 4 fields }	Object
_id	ObjectId("5ea186e8ad815bede96de952")	ObjectId
full_address	6401 University Ave Middleton, WI 53562	String
name	Crandalls Carryout & Catering	String
stars	4.0	Double
> (3) ObjectId("5ea186e8ad815bede96de95e")	{ 4 fields }	Object
> (4) ObjectId("5ea186e8ad815bede96de97b")	{ 4 fields }	Object
> (5) ObjectId("5ea186e8ad815bede96de98d")	{ 4 fields }	Object

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

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

Key	Value	Type
▼ (1) ObjectId("5ea2076a4c58e929bca6ac50")	{ 12 fields }	Object
_id	ObjectId("5ea2076a4c58e929bca6ac50")	ObjectId
yelping_since	2009-08	String
votes	{ 3 fields }	Object
funny	0	Int32
useful	0	Int32
cool	0	Int32
review_count	1	Int32
name	Bernard	String
user_id	xP3SPgfgW2vc5Zj5uV8SEA	String
friends	[0 elements]	Array
fans	0	Int32
average_stars	5.0	Double
type	user	String
compliments	{ 0 fields }	Object
elite	[0 elements]	Array
> (2) ObjectId("5ea2076c4c58e929bca7aa73")	{ 12 fields }	Object
> (3) ObjectId("5ea207694c58e929bca67867")	{ 12 fields }	Object

- e) Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (*tip*) w 2012. Wynik posortuj alfabetycznie według liczby (*tip*).

```
db.getCollection('tip').aggregate([
    {$match:{'date':/2012/}},
    {$group:{_id:"$business_id",
    tips:{$sum:1}}},
    {$sort:{tips:1}}])
```

Key	Value	Type
▼ (1) g1Rz8AZfcO2A97k6dfesow	{ 2 fields }	Object
_id	g1Rz8AZfcO2A97k6dfesow	String
tips	1.0	Double
▼ (2) N-gqleVDebl3qgwxf1jMcv	{ 2 fields }	Object
_id	N-gqleVDebl3qgwxf1jMcv	String
tips	1.0	Double
▼ (3) xWmfGEZ4msvVsASzIkW2Zw	{ 2 fields }	Object
_id	xWmfGEZ4msvVsASzIkW2Zw	String
tips	1.0	Double
▼ (4) zJ9J3KunQy14DB0vAxxOwQ	{ 2 fields }	Object
_id	zJ9J3KunQy14DB0vAxxOwQ	String
tips	1.0	Double
▼ (5) lBP22oiElBtqEblBvwq28g	{ 2 fields }	Object
_id	lBP22oiElBtqEblBvwq28g	String
tips	1.0	Double

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

```
db.getCollection('review').aggregate([
  {$group:{
    _id:"$business_id",
    avg_stars:{$avg:"$stars"}
  }},
  {$match:{
    avg_stars:{$gte:4.0}}}]])
```

Key	Value	Type
▼ (1) Ql-gKncMaGYEYw4n...	{ 2 fields }	Object
_id	Ql-gKncMaGYEYw4nouW2KQ	String
avg_stars	4.85714285714286	Double
▼ (2) MHTwKO1Udv4oF7s...	{ 2 fields }	Object
_id	MHTwKO1Udv4oF7sUM4pbfg	String
avg_stars	4.0	Double
▼ (3) OeZ04ZUJ1o7ia-rwi...	{ 2 fields }	Object
_id	OeZ04ZUJ1o7ia-rwiq7CiA	String
avg_stars	4.333333333333333	Double

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

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

0.116 sec.

Removed 1576 record(s) in 116ms

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

```
function addReview(user_id,review_id,text,business_id){
  db.getCollection('review').insert({
    votes:{
      funny:0,
      useful:0,
      cool:0},
    user_id:user_id,
    review_id:review_id,
    stars:0,
    date:new Date(),
    text:text,
    type:'review',
    business_id:business_id
  })
}

//Calling the function
addReview('zvJdCrp42y0ZrxKffwGQLA','zvJdCrp42y0ZrxKff48dw1','review','zvJdCrpjfk3k1xKff48dw1')
```



0.003 sec.

Inserted 1 record(s) in 4ms

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

```
function getBusinessOfCategory(category){
  return db.getCollection('business').find({categories:category})
}
getBusinessOfCategory('Mass Media')
```

Key	Value	Type
▼ (1) ObjectId("5ea186e8...")	{ 16 fields }	Object
_id	ObjectId("5ea186e8ad815bede...")	ObjectId
business_id	oLctHIA1AxmsgOuu4dM6Vw	String
full_address	4156 County Rd B Mc Farland, ...	String
> hours	{ 0 fields }	Object
open	false	Boolean
▼ categories	[2 elements]	Array
[0]	Television Stations	String
[1]	Mass Media	String
city	Mc Farland	String
review_count	10	Int32
name	Charter Communications	String
> neighborhoods	[0 elements]	Array
longitude	-89.3229199	Double
state	WI	String
stars	1.5	Double
latitude	42.9685074	Double
> attributes	{ 0 fields }	Object
type	business	String
> (2) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (3) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (4) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (5) ObjectId("5ea186e8...")	{ 16 { 16 fields } }	Object
> (6) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (7) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (8) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (9) ObjectId("5ea186e8...")	{ 16 fields }	Object
> (10) ObjectId("5ea186e8...")	{ 16 fields }	Object

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

Imie użytkownika - Tyler

Key	Value	Type
▼ (1) ObjectId("5ea20768...")	{ 12 fields }	Object
_id	ObjectId("5ea207684c58e929b...")	ObjectId
yelping_since	2011-08	String
> votes	{ 3 fields }	Object
review_count	6	Int32
name	Tyler	String
user_id	4dJLZvpYRcjQ6qDR5i7WsA	String
> friends	[0 elements]	Array
fans	0	Int32
average_stars	4.33	Double
type	user	String
> compliments	{ 0 fields }	Object
> elite	[0 elements]	Array

Tworzymy i wykonujemy skrypt:

```
function modifyUserNameByID(user_id,new_name){
  db.user.updateOne(
    {user_id:user_id},
    {$set:{
      name:new_name}})
}
modifyUserNameByID('4dJLZvpYRcjQ6qDR5i7WsA','Andrii')
```

Wynik:

user 0.001 sec.		
Key	Value	Type
▼ (1) ObjectId("5ea20768...")	{ 12 fields }	Object
_id	ObjectId("5ea207684c58e929b...")	ObjectId
yelping_since	2011-08	String
> votes	{ 3 fields }	Object
review_count	6	Int32
name	Andrii	String
user_id	4dJLZvpYRcjQ6qDR5i7WsA	String
> friends	[0 elements]	Array
fans	0	Int32
average_stars	4.33	Double
type	user	String
> compliments	{ 0 fields }	Object
> elite	[0 elements]	Array

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

```
var map_fun = function(){
    emit(this.business_id,this.likes);
}
var reduce_fun = function(business_id,likes){
    return Array.avg(likes);
}
db.getCollection('tip').
    mapReduce(
        map_fun,
        reduce_fun,
        {out:"averageBusinessTip"}
    )
db.averageBusinessTip.find()
```

Key	Value	Type
▼ (1) --1emggGHgoG6ipd...	{ 2 fields }	Object
_id	--1emggGHgoG6ipd_RMb-g	String
value	0.0	Double
▼ (2) --5jkZ3-nUPZxUvtcbr...	{ 2 fields }	Object
_id	--5jkZ3-nUPZxUvtcbr8Uw	String
value	0.0625	Double
▼ (3) --BlvDO_RG2yEIKu9X...	{ 2 fields }	Object
_id	--BlvDO_RG2yEIKu9XA1_g	String
value	0.0	Double
▼ (4) --DI2rW_xO8GuYBo...	{ 2 fields }	Object
_id	--DI2rW_xO8GuYBomlg9zw	String
value	0.0	Double
▼ (5) --Y_2IDOtVDioX5bw...	{ 2 fields }	Object
_id	--Y_2IDOtVDioX5bwF6Glw	String
value	0.0	Double
▼ (6) --jFTZmywe7StuZ2h...	{ 2 fields }	Object
_id	--jFTZmywe7StuZ2hEjxyA	String
value	0.0	Double
▼ (7) --qeSYxyn62mMjWv...	{ 2 fields }	Object
_id	--qeSYxyn62mMjWvznNTdg	String
value	0.0	Double

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

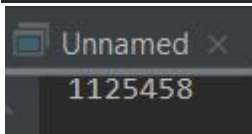
- a. Zwróć bez powtórzeń wszystkie nazwy miast w których znajdują się firmy (*business*).
Wynik posortuj na podstawie nazwy miasta alfabetycznie.

```
private void getSortedBusinessTowns(){
    List<String>
towns=db.getCollection("business").distinct("city",String.class).into(new
ArrayList<>());
    Collections.sort(towns);
    for(String tmp:towns){
        System.out.println(tmp);
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_192\bin\java
Ahwatukee
Anthem
Apache Junction
Arcadia
Atlanta
Avondale
Black Canyon City
Bonnyrigg
Boulder City
Buckeye
C Las Vegas
Cambridge
Carefree
Casa Grande
Cave Creek
Centennial Hills
Central City Village
Central Henderson
Chandler
```


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

```
private void reviewsIn2011Number(){
    System.out.println(db.getCollection("review").countDocuments(new
    Document("date",new Document("$gte","2001-01-01"))));
}
```



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

```
private void closedFirmsInfo() {
    List<Document> firms = db.getCollection("business").
        find(eq("open", false)).
        projection(fields(include("name", "stars", "full_address"),
        excludeId()))).
        into(new ArrayList<>());
    for (Document tmp : firms) {
        System.out.println(tmp);
    }
}
```

```
Las Vegas, NV 89123, name=Vid Noodles, stars=4.0}}
Document{{full_address=10430 S Eastern Ave
Anthem
Henderson, NV 89052, name=Carmine's, stars=3.0}}
Document{{full_address=3000 W Ann Rd
Ste 109
North Las Vegas, NV 89031, name=Super No1 Chinese Restaurant, stars=3.5}}
Document{{full_address=13014 N Saguaro Blvd
```

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

```
private void usersWithoutFunnyOrUseful(){
    List<Document> users=
    db.getCollection("user").find(or(eq("votes.funny",0),eq("votes.useful",0))
    ).sort(ascending("name")).into(new ArrayList<>());
    for (Document tmp : users) {
        System.out.println(tmp);  }}
}
```

```
Document({_id=5ea2076d4c58e929bca836c6, yelping_since=2011-03, votes=Document({funny=0, usefu1=0, cool=0}), review_count=2, name=chad, user_id=2Gz0KzCqHheKD1PIkx9jw, fr
Document({_id=5ea207694c58e929bca8046, yelping_since=2011-09, votes=Document({funny=0, usefu1=0, cool=0}), review_count=2, name=chantal, user_id=8b_xOerDwJ0aUW5qLBdMkw,
Document({_id=5ea207694c58e929bca668ed, yelping_since=2010-06, votes=Document({funny=0, usefu1=0, cool=0}), review_count=9, name=charise, user_id=cG6-toQ5sNeGKDZ0n-K0JA,
Document({_id=5ea207704c58e929bca9a4ba, yelping_since=2011-03, votes=Document({funny=0, usefu1=0, cool=0}), review_count=2, name=charlee, user_id=26s3HYFFU7wvIsPZYRF-pA,
Document({_id=5ea2076d4c58e929bca80c85, yelping_since=2010-10, votes=Document({funny=0, usefu1=1, cool=0}), review_count=3, name=charlesb, user_id=x737irkRuS8ABGRc2ILLAA
Document({_id=5ea2076f4c58e929bca91824, yelping_since=2011-05, votes=Document({funny=0, usefu1=1, cool=0}), review_count=4, name=charlotte, user_id=UELgRj3bQhKfMqDgUoMXl3
Document({_id=5ea207684c58e929bca5ed58, yelping_since=2011-11, votes=Document({funny=0, usefu1=1, cool=1}), review_count=4, name=charmen, user_id=AmwQ5M5y78GtMAYi1J4hkA,
Document({_id=5ea2076e4c58e929bca8b35e, yelping_since=2011-02, votes=Document({funny=0, usefu1=1, cool=1}), review_count=1, name=chastity, user_id=Qzyyv5Cs96wHv1Hl_O_cVw
Document({_id=5ea207684c58e929bca5f3a3, yelping_since=2011-02, votes=Document({funny=0, usefu1=3, cool=0}), review_count=2, name=chaz, user_id=pGVvobQfj1mkV-9N1Kufqg, fr
Document({_id=5ea2076f4c58e929bca91618, yelping_since=2010-06, votes=Document({funny=0, usefu1=8, cool=1}), review_count=5, name=chaz, user_id=9_cDp0KeB55She2001IXTXQ, fr
Document({_id=5ea2076e4c58e929bca88995, yelping_since=2011-03, votes=Document({funny=0, usefu1=0, cool=0}), review_count=2, name=che, user_id=YwuhgrKeTPm7ZBF05Tfpgg, fri
Document({_id=5ea207694c58e929bca63644, yelping_since=2011-06, votes=Document({funny=1, usefu1=0, cool=0}), review_count=3, name=chef Jay, user_id=h_3dLHZGESIA5bT1TihDrA
Document({_id=5ea207694c58e929bca67e28, yelping_since=2010-01, votes=Document({funny=0, usefu1=0, cool=0}), review_count=2, name=chelsea, user_id=WjW0aKh3QHbsf_met00KUw,
Document({_id=5ea2076a4c58e929bca6e9d9, yelping_since=2011-08, votes=Document({funny=0, usefu1=1, cool=0}), review_count=2, name=chelsey, user_id=5p7N0s10VVnZQ2L_570hQ,
Document({_id=5ea2076f4c58e929bca98cf3, yelping_since=2011-08, votes=Document({funny=1, usefu1=0, cool=0}), review_count=3, name=cherie, user_id=GciYfict2mNU_0TovJwSg,
Document({_id=5ea2076e4c58e929bca8c2d0, yelping_since=2011-08, votes=Document({funny=0, usefu1=0, cool=0}), review_count=1, name=cherisse, user_id=3CEFP9WkEgoIw81w7yJudDg
Document({_id=5ea2076e4c58e929bca89729, yelping_since=2007-12, votes=Document({funny=0, usefu1=2, cool=1}), review_count=6, name=cherry, user_id=LrRGrlJswFbI3Om7bqgXYA,
Document({_id=5ea207684c58e929bca5f850, yelping_since=2008-03, votes=Document({funny=0, usefu1=0, cool=0}), review_count=3, name=cheryl, user_id=OY6Vcv25fwivg4zki1F4hA, fr
Document({_id=5ea207714c58e929bca9c4e1, yelping_since=2009-01, votes=Document({funny=0, usefu1=0, cool=1}), review_count=14, name=cheryl, user_id=J6bF6kHsx0_kQNUras8-Cw,
Document({_id=5ea2076f4c58e929bca91c20, yelping_since=2009-03, votes=Document({funny=0, usefu1=0, cool=0}), review_count=1, name=chinda, user_id=1ae18HMFpJQYPE8LUF6EDQ,
Document({_id=5ea2076d4c58e929bca84547, yelping_since=2009-03, votes=Document({funny=0, usefu1=0, cool=1}), review_count=2, name=ching, user_id=hOV-ot-1Hw-unP4ZYr4ETw, fr
Document({_id=5ea207684c58e929bca5f811, yelping_since=2007-10, votes=Document({funny=0, usefu1=0, cool=0}), review_count=1, name=chip, user_id=FHYSGZ8PaRJQUNJi1vml8A, fr
Document({_id=5ea207694c58e929bca6284c, yelping_since=2011-07, votes=Document({funny=0, usefu1=2, cool=0}), review_count=2, name=chris, user_id=qjKJlvoFud-LaeRRVXH-w, fr
Document({_id=5ea207694c58e929bca62924, yelping_since=2011-11, votes=Document({funny=0, usefu1=8, cool=2}), review_count=9, name=chris, user_id=yLMG62U9-Vot8a4Sun0wzQ, f
```

- e. Określ, ile każde przedsiębiorstwo otrzymało wskazówek/napiwków (*tip*) w 2012. Wynik posortuj alfabetycznie według liczby (*tip*).

```
private void businessTipNumber(){
    List<Document>
    firms=db.getCollection("tip").aggregate(Collections.singletonList(Aggregates.gro
up("$business_id", Accumulators.sum("tips",1))).into(new ArrayList<>());
    for (Document tmp : firms) {
        System.out.println(tmp);
    }
}
```

```
Document({_id=jKSYw_wJz3MKBVcXWAOYlQ, tips=12})
Document({_id=0YrT9iR0J3406fNKEsFJjg, tips=3})
Document({_id=lCCa5gxqRz3l4t4N9RJfJw, tips=5})
Document({_id=gV2APG0WE-pqR8i-ZBTDew, tips=1})
Document({_id=OrfxjpJXpK6WE9SaW3EU8Q, tips=7})
Document({_id=CCOyMTsbUm6Wm2fC_F2FIQ, tips=7})
Document({_id=FLegS3ljM7Wy9mxtA9BS4w, tips=1})
Document({_id=FcmegKpfN3-URLKGgbSyKg, tips=5})
Document({_id=jVz5XroGCuEBCa9V4_NNtQ, tips=38})
Document({_id=73hgZ8PLt4AKRhY3ATMSow, tips=6})
Document({_id=RO1fBofbm17ygeY2AiWfYg, tips=47})
Document({_id=plRQRyL_xAY8sUTm4viP6Q, tips=57})
Document({_id=PqjwKYHU_eHjw0q0N1ULJw, tips=3})
Document({_id=s0SBkMRdPQEntTzOB-RJmw, tips=1})
Document({_id=CocqwthulO8P1_m0Dr7PVg, tips=2})
Document({_id=uRmw4rk61El02tw95Kzu0w, tips=2})
Document({_id=2a-RVYQA7eIsyxcUbnFXsA, tips=6})
Document({_id=MxXc_978mY6sfoEHHInvDA, tips=1})
Document({_id=f08QL3vPDZVZVvJUsC-ZBw, tips=6})
Document({_id=LTHq7A5vVAwtqqu5i2ZLUQ, tips=18})
Document({_id=XbIzDzo_Uc0kuDMU0H2QjQ, tips=3})
Document({_id=czoXs1z2MGzPXNP3W1d0Yw, tips=5})
Document({_id=ipMMVBGEWABP4PIfudvRVw, tips=8})
```

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

```
private void allFirmsThatHasAverageReviewNumberGraterThanFour(){
    List<Document> firms=db.getCollection("review").aggregate(Arrays
        .asList(Aggregates
            .group("$business_id",Accumulators.avg("avg_rate","$stars")),Aggregates.match(Fi
            lters.gte("avg_rate",4.0))))).into(new ArrayList<>());
    for (Document tmp : firms) {
        System.out.println(tmp);
    }
}
```

```
Document{{_id=LK1-GXhu_bmOhvhoFVldVg, avg_rate=5.0}}
Document{{_id=SmX_sTfTI4Hhyj8Rbf9teg, avg_rate=4.7272727272727275}}
Document{{_id=sa-CgzlC0m5xrspvBbz6xA, avg_rate=4.107142857142857}}
Document{{_id=l1zV_lqnpWeRUGbm4d3r1w, avg_rate=4.333333333333333}}
Document{{_id=fUFykr8Mf9VANDHAFeo_Rw, avg_rate=4.529411764705882}}
Document{{_id=LGYS0d7lnhwAGuFJOmo6Rw, avg_rate=4.5}}
Document{{_id=V8QNpf_QtpTT5IEoILUKCA, avg_rate=4.5}}
Document{{_id=zGzr1Bp2Gby6aJtt0jqr-g, avg_rate=5.0}}
Document{{_id=PHWH6_Tv09Q-7pF8b75HFQ, avg_rate=5.0}}
Document{{_id=WF_--gu5Z9WB3CU2kJQ1fg, avg_rate=4.230769230769231}}
Document{{_id=0YrT9iR0J3406FNKEsFJjg, avg_rate=4.125}}
Document{{_id=lCCa5gxqRz3l4t4N9RJfJw, avg_rate=4.451612903225806}}
Document{{_id=gV2APG0WE-pqR8i-ZBTDew, avg_rate=4.0}}
Document{{_id=OrfxjpJXpK6WE9Saw3EU8Q, avg_rate=4.055555555555555}}
Document{{_id=LFszpBj9_chk-q7K0v22bA, avg_rate=4.833333333333333}}
Document{{_id=CC0yMTSbUm6Wm2fC_F2FIQ, avg_rate=4.346153846153846}}
Document{{_id=JDwVex4yni5pLVhFeCYH3w, avg_rate=5.0}}
Document{{_id=NOscNsgq00__WtFUSQCZrA, avg_rate=5.0}}
Document{{_id=PwlqT9PpJ-zJs5R271pAmA, avg_rate=4.714285714285714}}
Document{{_id=73hgZ8PLt4AKRhY3ATMSow, avg_rate=4.2}}
Document{{_id=C0VBu6tkY551KEuIWwf1KQ, avg_rate=5.0}}
Document{{_id=PqJwKYHU_eHjw0q0N1ULJw, avg_rate=4.03125}}
Document{{_id=hV-FzNjNznWCgnq8a0mmiQ, avg_rate=4.5}}
Document{{_id=ONYFR0gFZt7A2C3kjfgErg, avg_rate=4.0}}
Document{{_id=w5CX5GRNwZ5ktzDMsdxRfw, avg_rate=4.0}}
Document{{_id=MxXc_978mY6sfoEHHInvDA, avg_rate=4.363636363636363}}
Document{{_id=f08QL3vPDZWVvjUsC-ZBw, avg_rate=4.0}}
Document{{_id=ipMMVBGEWABP4PIfudvRVw, avg_rate=4.666666666666667}}
```

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

```
private void deleteAllRateTwoZero(){
    db.getCollection("business").deleteMany(eq("stars",2.0));
}
```

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

Przy pomocy generatora danych napisanego w Python została wygenerowana baza która składa się z trzech kolekcji **user** która przechowuje stworzonych użytkowników, **offers** to są produkty którzy można kupić i **orders** lista zakupów użytkownika.

```
from pymongo import MongoClient
from faker import Factory
from typing import List
from random import randint
import time

client = MongoClient("mongodb://localhost:27017")
db = client.eSklep

class Comment(object):
    def __init__(self, mark: int, review: str, senderUID: str):
        self.mark = mark
        self.review = review
        self.senderUID = senderUID

def create_users(fake):
    genName = fake.first_name()
    genSurname = fake.last_name()
    role = "user"
    password = fake.pystr(8, 10)
    deliveryAddress = fake.street_address()
    phoneNumber = fake.phone_number()
    email = fake.ascii_free_email()
    result = db.users.insert_one(
        {
            'password': password,
            'email': email,
            "phoneNumber": phoneNumber,
            "deliveryAddress": deliveryAddress,
            'name': genName,
            'surname': genSurname,
            'role': role
        }
    )
    print("User added")

def create_offers(fake):
    title = fake.text(10)
    image = fake.image_url()
    sellerUID = fake.random_element([*(str(id) for id in
db.users.find().distinct('_id'))])
    rating = fake.random_element([1, 2, 3, 4, 5])
    price = fake.random_int(15, 300)
    description = fake.paragraph(2)
    comments = []
```

```

numberOfComments = fake.random_element([1, 2, 3, 4])
for i in range(0, numberOfComments):
    mark = fake.random_element([1, 2, 3, 4, 5])
    review = fake.sentence(10)
    senderUID = fake.random_element([*(str(id) for id in
db.users.find().distinct('_id'))])
    comment = Comment(mark, review, senderUID)
    comments.append(comment.__dict__)
result = db.offers.insert_one(
    {
        'title': title,
        'rating': rating,
        'sellerUID': sellerUID,
        'price': price,
        'description': description,
        'imageSrc': image,
        'comments': comments

    }
)
print("Offer added")

def create_orders(fake):
    date = str(fake.date_this_decade())
    offerUID = fake.random_element([*(str(id) for id in
db.users.find().distinct('_id'))])
    result = db.orders.insert_one(
        {
            'date': date,
            'offerUID': offerUID

        }
    )

    print("Order added")

def generate(fake):
    create_offers(fake)
    create_users(fake)
    create_orders(fake)

fake = Factory.create()
generate(fake)

```

Struktura Users


```

/* 1 */
{
  "_id" : ObjectId("5e9066e19971958803721caf"),
  "password" : "admin",
  "email" : "admin@gmail.com",
  "phoneNumber" : "+48554823445",
  "deliveryAddress" : "Kraków,Witolda Budryka 1",
  "name" : "John",
  "surname" : "Smith",
  "role" : "admin"
}
/* 2 */

```

Struktura Offers

```

{
  "_id" : ObjectId("5e91f29e40377225249caf91"),
  "title" : "Green stuff",
  "rating" : 4,
  "sellerUID" : "5e91f74140377225249caf93",
  "price" : "15",
  "description" : "Green tea contains healthy bioactive compounds.",
  "imgSrc" : "https://www.bbcgoodfood.com/sites/default/files/guide/guide-image/2017/01/green-tea.jpg",
  "comments" : [
    {
      "mark" : "4",
      "review" : "Pretty good tea for that price",
      "senderUID" : "5e9066e19971958803721caf"
    }
  ]
}

```

Struktura Orders

```

/* 1 */
{
  "_id" : ObjectId("5e9213a773437fe7ed319f77"),
  "date" : "2020-02-13",
  "offerUID" : "5e9210aa6621cfef554145c7"
}

```

Dana struktura pozwala na wygodne przechowywanie dotyczące klientów, zakupu oraz przedmiotu zakupu.