

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

```
1 use PawełKocimskiA;
2 db.getCollection("business").distinct("city").sort();
3
```

Text	Array	Text	Array ×
[] Array			
[Index]			
0	" Ahwatukee		
1	" Anthem		
2	" Apache Junction		
3	" Arcadia		
4	" Atlanta		
5	" Avondale		
6	" Black Canyon City		
7	" Bonnyrigg		
8	" Boulder City		
9	" Buckeye		
10	" C Las Vegas		
11	" Cambridge		
12	" Carefree		
13	" Casa Grande		
14	" Cave Creek		

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

```
1 use PawełKocimskiA;
2 db.review.find({
3   "date" : {"$gte": "2011-01-01"}}).count()
```

Text	Find	Text	Find	Text	Text	Text
1	880318					
2						

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

```
1 use PawełKocimskiA;
2 db.getCollection("business").find(
3   {
4     "open" : false
5   },
6   {
7     name: 1, full_address: 1, stars: 1, _id: 0
8   }
9 );
10
```

full_address	name	stars
6401 University	Crandalls Carr	4.0
6230 University	Mi Cocina	3.0
4156 County Rd	Charter Comm	1.5
6625 Century A	Stamm House	2.0
6661 University	Tangles	2.0
1901 Cayuga St	Soup Factory	3.0
1632 W Main St	Deli Roma	4.0
2910 E Washing	Java Detour	4.5
4120 Monona E	Bongo Video	5.0
3001 N Sherma	Rocky Rococo	1.5
1941 Winneba	Designs By the	2.5
3729 E Washing	Williamson Bik	3.5
1902 E Johnso	Area 51 Vintag	3.0
1151 N Sherma	Dorn True Valu	2.5
3044 Atwood	Wagon's Garden	2.5

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

```

1 use PawełKocińskiA;
2 db.getCollection("user").find(
3 {
4   "$or" : [
5     {
6       "votes.useful" : NumberLong(0)
7     },
8     {
9       "votes.funny" : NumberLong(0)
10    }
11  ]
12 }
13 ).sort({"name": 1});

```

elping_since	votes	review_count	name	user_id	friends	fans	average_stars	type	compliments	elite
2012-03	{ 3 fields }	2	Brandon	CFWeBCqVvHz	[1 elements]	0	4.5	user	{ 0 fields }	{ 0 elements }
2011-10	{ 3 fields }	5	David	MSIQGr6omqc	[33 elements]	0	4.6	user	{ 1 fields }	{ 0 elements }
2011-04	{ 3 fields }	6	Jenelle	U5yKFJdqKSTF	[0 elements]	0	5.0	user	{ 0 fields }	{ 0 elements }
2013-11	{ 3 fields }	4	Persian	bSoPY5UR-fXS	[0 elements]	0	4.0	user	{ 0 fields }	{ 0 elements }
2013-03	{ 3 fields }	4	Virydiana	wP-nekfpFZEZ	[5 elements]	1	4.5	user	{ 4 fields }	{ 0 elements }
2013-03	{ 3 fields }	1	Maria	Os5f3TNpM7_	[0 elements]	0	5.0	user	{ 0 fields }	{ 0 elements }
2009-06	{ 3 fields }	10	rick	28s2JRyWuRG	[0 elements]	0	2.67	user	{ 1 fields }	{ 0 elements }
2014-06	{ 3 fields }	3	006969123	FLtId0SF_puLC	[0 elements]	0	1.0	user	{ 0 fields }	{ 0 elements }
2012-10	{ 3 fields }	2	A	MjLcGNLsSUz	[0 elements]	0	3.0	user	{ 0 fields }	{ 0 elements }
2013-01	{ 3 fields }	2	A	aHuT-eiSBLORl	[0 elements]	0	3.0	user	{ 0 fields }	{ 0 elements }
2013-01	{ 3 fields }	1	A	kanJLBDaadQ	[0 elements]	0	5.0	user	{ 0 fields }	{ 0 elements }
2012-01	{ 3 fields }	8	A	UuV8Iz90PdLn	[0 elements]	0	2.5	user	{ 0 fields }	{ 0 elements }
2012-01	{ 3 fields }	1	A	ArZiH7p8lIMn	[0 elements]	0	5.0	user	{ 0 fields }	{ 0 elements }
2013-03	{ 3 fields }	9	A	SUmjgKXWVh	[0 elements]	1	4.33	user	{ 0 fields }	{ 0 elements }

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

```

1 use PawełKocińskiA;
2 db.getCollection("tip").aggregate(
3 {
4   "$match" : {
5     "date" : {
6       "$gte" : "2012-01-01",
7       "$lte" : "2012-12-31"
8     }
9   },
10  {
11    "$group" : {
12      "id" : "$business_id",
13      "count" : {
14        "$sum" : 1.0
15      }
16    }
17  },
18  {
19    "$sort" : {
20      "count" : 1
21    }
22  }
23 }
24 );

```

_id	count
CbUkI9Bpdl_m	1.0
ToSm1CH3xga	1.0
uTVf-raBw4J9k	1.0
FHJQZ0p5ByTi	1.0
MHUX3HRj-gt	1.0
fScnL-o3j2G9gl	1.0
hwJJPXF0eOhl	1.0
JoU_SO4RAHY	1.0
DJg1loBuRmA	1.0

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

Aggregate Aggregate Aggregate x

50 Documents 1 to 50

Result> avg_stars

_id	avg_stars
BVxlrYWgmi-8	4.0
BMjggIgOghBM	5.0
iyKyJoDcbkGr	5.0
yZXEELxi8KMw	4.666666666666667
UH4BReVAqpN	4.0
qZkW6s0qxtU	4.333333333333333
ONwda6h7mRf	5.0
jTUe25wOiM0z	5.0
0EXVIF07vw-Q	4.75
pOz8fQ1Y8fP3	5.0
uV2EhAuAmPE	4.666666666666667
NyHw4bpksLTe	5.0
VNfCMuY4vnq	4.0
M3G_xbxxfkup	4.8
dtgFizN4sWDp	5.0

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

```
1 try {
2   db.business.deleteMany(
3     { "stars" : { $eq: 2 } }
4   );
5 } catch (e) {
6   print(e);
7 }
```

Document x

50 Documents 1 to 1

Result> acknowledged

_id	acknowledged	deletedCount
	true	1576.0

- 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("zvJCcprpm2y0ZrxKffwGQLA", "ebcN2aqmNUuYNoYvQErnA", "Description of the inserted review", "vcNAWiLM4dR7D2mwJ7nCA")
```

0.085 sec.

Inserted 1 record(s) in 85ms

- 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 getBusinessByCategory(category){
2   return db.business.find({
3     categories: category
4   })
5 }
6
7 getBusinessByCategory("Restaurants")
```

Documents 1 to 20

_id	business_id	full_address	hours	open	categories	city	review_count	name	neighborhoods	longitude
5e7a337f92be...	rdAdANPNOC...	4412 Siggelkov	[7 fields]	true	[2 elements]	Mc Farland	33.0	Green Lantern	[0 elements]	-89.3061344
5e7a337f92be...	_wZTYYL7cuta	4506 Larson Br	[0 fields]	true	[5 elements]	Mc Farland	31.0	Beach House R	[0 elements]	-89.303789
5e7a337f92be...	zOc8lbyViUZajl	5813 Main St	[5 fields]	true	[2 elements]	Mc Farland	4.0	Spartan Pizza	[0 elements]	-89.288567
5e7a337f92be...	UgJVZTSOaYoE	4850 Larson Br	[0 fields]	true	[2 elements]	Mc Farland	8.0	Main Moon Chi	[0 elements]	-89.2986278
5e7a337f92be...	77ESrCo7hQ9e	6230 Universit	[7 fields]	false	[2 elements]	Middleton	17.0	Mi Cocina	[0 elements]	-89.4874867
5e7a337f92be...	SKLw05kElIZcp	2039 Allen Blvc	[7 fields]	true	[5 elements]	Middleton	41.0	Imperial Garde	[0 elements]	-89.485169
5e7a337f92be...	JwUE5GmEO-s	6162 US Highw	[0 fields]	true	[1 elements]	De Forest	26.0	Pine Cone Rest	[0 elements]	-89.335844
5e7a337f92be...	KTqNU4pLO23i	2411 Allen Blvc	[0 fields]	true	[4 elements]	Middleton	3.0	Domino's Pizz	[0 elements]	-89.4862189
5e7a337f92be...	uCykseHzYSx	505 W North Si	[7 fields]	true	[2 elements]	De Forest	16.0	Deforest Famil	[0 elements]	-89.353437
5e7a337f92be...	MKsb2VpLB-0l	6913 Universit	[0 fields]	true	[2 elements]	Middleton	12.0	China Wok Buf	[0 elements]	-89.5006856
5e7a337f92be...	ShEYKerTwb2L	6720 Frank Llo	[7 fields]	true	[5 elements]	Middleton	38.0	Prairie Cafe &	[0 elements]	-89.496995
5e7a337f92be...	HaBkxSPwvbB	2401 Parment	[7 fields]	true	[4 elements]	Middleton	11.0	Paul's Neighbc	[0 elements]	-89.5111616
5e7a337f92be...	KW6HejC-67KS	8414 Old Sauk	[7 fields]	true	[2 elements]	Middleton	28.0	Chin's Asia Fre	[0 elements]	-89.52848018
5e7a337f92be...	wL-7A4jCOF27N	2608 Allen Blvc	[7 fields]	true	[2 elements]	Middleton	6.0	Grand China R	[0 elements]	-89.4864907

- 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, changedName){
2     db.user.update(
3         { _id: new ObjectId(id)},
4         {
5             $set:{ name: changedName}
6         }
7     })
8 }
9
10 modifyUser("5e7a34c192be9b08d7007aa2", "Paweł Kocimski")
11 db.user.find({ _id: new ObjectId("5e7a34c192be9b08d7007aa2")})
```

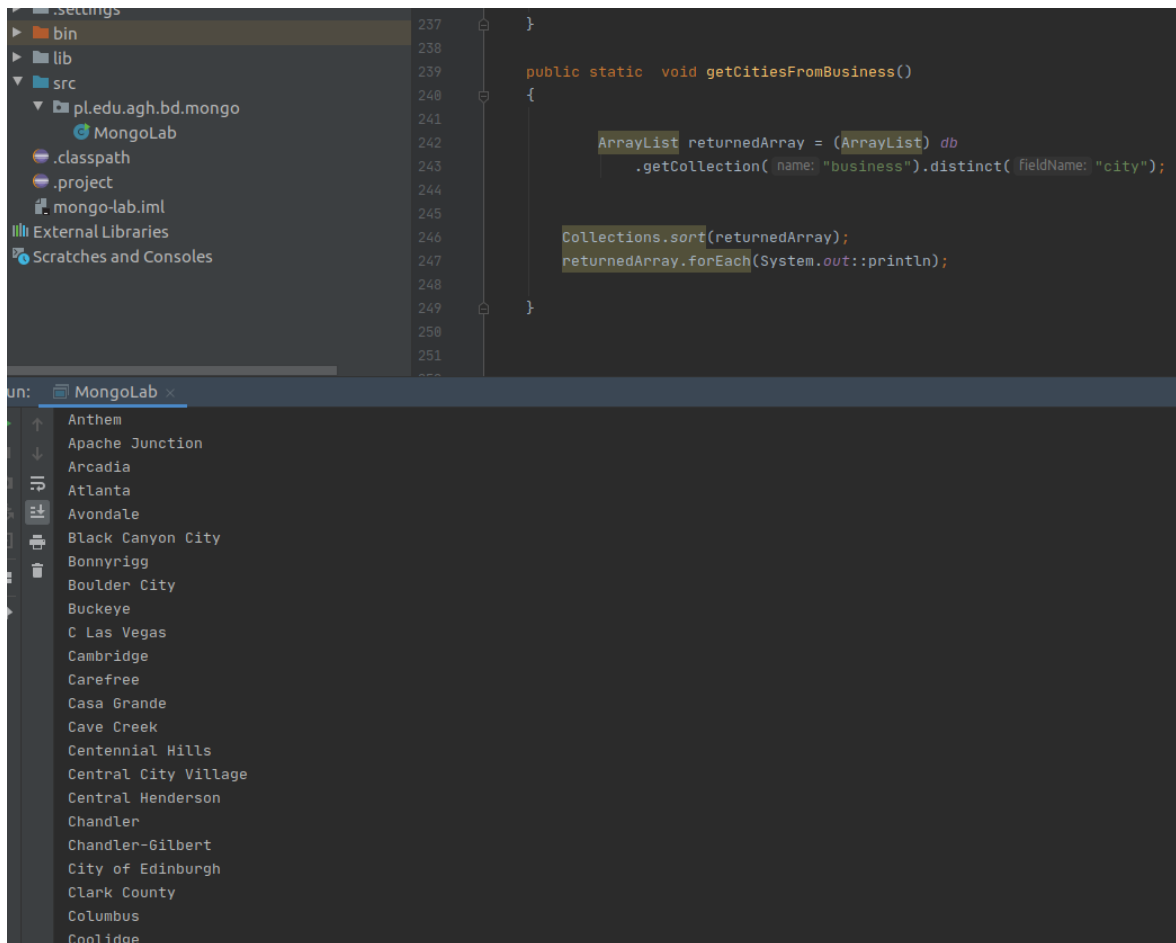
Document	Document	Document	Document	Document
Documents 1 to 1				
<pre>1 { 2 "_id" : ObjectId("5e7a34c192be9b08d7007aa2"), 3 "yelping_since" : "2010-04", 4 "votes" : { 5 "funny" : 23.0, 6 "useful" : 28.0, 7 "cool" : 18.0 8 }, 9 "review_count" : 38.0, 10 "name" : "Paweł Kocimski", 11 "user_id" : "BwvLE7SguUtXP-InnJbjSw", 12 "friends" : [13 "EVWnjbS4hVbspI0M4YH4Pw", 14 "acuoEIpGPqKwvFxFtDRVg", 15 "8LYEPuIG-ltZro0oEBREdw", 16 "lh6VX3J1FCefn4yDJuydqg", 17 "F0fUxCjxD5xpKM_7mphLJg", 18 "N7DxZbMcCP0FSD9ISUzQJw", 19 "qjDy5Kq_zN4bcxVIXt7BCA", 20 "4wpYB6Qd5YidLC_X0jYB9w", 21 "bhSlsx898TzpPCKBrgt9eQ", 22 "Chsa9RusRaejI48p6eySCQ", 23 "JFAdTlFkurs2xxBtG8xXYQ", 24 "xNjpfbbbhq4F2JREBdAxLg", 25 "xGuh7FY51oUI_20qqYo4PQ",</pre>				

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

```
1 var mapFunction1 = function() {
2   var key = this.business_id;
3   var value = {count: 1};
4   emit(key, value);
5 };
6
7 var reduceFunction1 = function(keyBusId, countTips){
8   var counter = 0;
9   countTips.forEach(function(val){counter+=val.count;})
10  return counter;
11 };
12
13 var finalizeFunction = function(key, reducedVal)
14 {
15   return reducedVal;
16 }
17
18 db.tip.mapReduce(
19   mapFunction1,
20   reduceFunction1,
21   { out: "map_reduce_avg_tips",
22     finalize: finalizeFunction
23 }
24 )
25
26 db.map_reduce_avg_tips.find()
```

Find	Find	Find	Find	Document	Find	Te
<div>Navigation: 50 Documents 1 to 50</div> <pre>1 { 2 "_id" : "--1emggGHgoG6ipd_RMb-g", 3 "value" : 6.0 4 } 5 { 6 "_id" : "--5jkZ3-nUPZxUvtcbr8Uw", 7 "value" : 16.0 8 } 9 { 10 "_id" : "--BlvD0_RG2yElKu9XA1_g", 11 "value" : 21.0 12 } 13 { 14 "_id" : "--Dl2rW_x08GuYBomlg9zw", 15 "value" : 2.0 16 } 17 { 18 "_id" : "--M3p0uPjvE1_F5ST-"</pre>						

- 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.



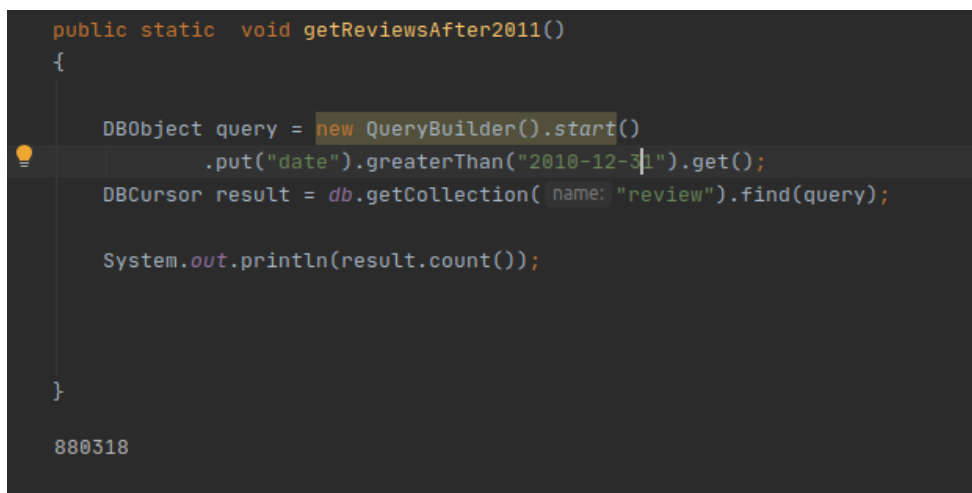
The screenshot shows an IDE with a project named 'MongoLab'. The code in the editor is as follows:

```
237 }
238
239 public static void getCitiesFromBusiness()
240 {
241     ArrayList returnedArray = (ArrayList) db
242         .getCollection( name: "business").distinct( fieldName: "city");
243
244     Collections.sort(returnedArray);
245     returnedArray.forEach(System.out::println);
246 }
247
248
249
250
251
```

The output window shows the following list of cities:

```
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
Chandler-Gilbert
City of Edinburgh
Clark County
Columbus
Coolidge
```

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



The screenshot shows an IDE with the following Java code:

```
public static void getReviewsAfter2011()
{
    DBObject query = new QueryBuilder().start()
        .put("date").greaterThan("2010-12-31").get();
    DBCursor result = db.getCollection( name: "review").find(query);

    System.out.println(result.count());
}
```

The output window shows the number 880318.

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

```
public static List<Document> getClosedNameAddressStars()
{
    BasicDBObject whereQuery = new BasicDBObject();
    whereQuery.put("open", false);

    return db.getCollection( name: "business").find(whereQuery)
        .projection(fields(include( ...fieldNames: "name", "full_address", "stars"), excludeId()))
        .into(new ArrayList<>()).forEach(System.out::println);
}
```

```
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 Hells, stars=5.0}}
```

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

```
public static DBCursor getUsersNegativeVotes()
{
    Bson filter = or(
        gt( fieldName: "votes.funny", value: 0),
        gt( fieldName: "votes.useful", value: 0),
        gt( fieldName: "votes.cool", value: 0)
    );

    db.getCollection( name: "user").createIndex((DBObject) ascending( ...fieldNames: "name"));

    return db
        .getCollection( name: "user")
        .find((DBObject) filter)
        .sort((DBObject)ascending( ...fieldNames: "name"));
}
```

```
{ "_id": {"$oid": "5e7a34c492be9b88d7845724"}, "yelping_since": "2013-12", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 11, "name": "L", "user_id": "6oDapua2Lz1f8I3-IDQNO", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d7845725"}, "yelping_since": "2012-09", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 5, "name": "Chris", "user_id": "LkhjK25mP_nL1TKQyvrAMA", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d7845727"}, "yelping_since": "2012-02", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 7, "name": "Michael", "user_id": "5mM_SM4cwNN4Yw1hAHL8w", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d7845729"}, "yelping_since": "2013-10", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 3, "name": "Marco", "user_id": "MMbnzw0QAhVt02R476NpnA", "friends": [{"aZyP3l6E",
{"_id": {"$oid": "5e7a34c492be9b88d784572b"}, "yelping_since": "2010-11", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "Sari", "user_id": "0PZ-laog6osQF8fC5ShXw", "friends": [{"ZT-X6uq5-h",
{"_id": {"$oid": "5e7a34c492be9b88d784572c"}, "yelping_since": "2011-01", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 6, "name": "Lee", "user_id": "Ndh-qCF5wA1g_mS30o0Q7w", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d784572f"}, "yelping_since": "2013-10", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 8, "name": "Eric", "user_id": "s2_P2w9-X0_K900CptAFeg", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d784573a"}, "yelping_since": "2013-07", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "Brian", "user_id": "l15VidsuHTsjAnQ5eTislu", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d7845737"}, "yelping_since": "2009-09", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 3, "name": "Beth", "user_id": "6avFMSwV1fZh131EIslnw", "friends": [{"bY43hsLRkQ",
{"_id": {"$oid": "5e7a34c492be9b88d7845739"}, "yelping_since": "2013-03", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "Ward", "user_id": "mpWwIeolJaagZ4j6kxySjw", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d784573b"}, "yelping_since": "2010-11", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 5, "name": "Breeze", "user_id": "Hx2q6aNBcD68ud7j0pDzA", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d784573c"}, "yelping_since": "2009-02", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 2, "name": "Nick1", "user_id": "V3KoITwFMQU1Da45Up68Yw", "friends": [{"L5J48U4ack",
{"_id": {"$oid": "5e7a34c492be9b88d7845743"}, "yelping_since": "2013-01", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "Chuck", "user_id": "CwTKfRy1N2Z_M35pWzds1w", "friends": [], "fans": 0,
{"_id": {"$oid": "5e7a34c492be9b88d7845744"}, "yelping_since": "2014-01", "votes": {"funny": 0, "useful": 0, "cool": 0}, "review_count": 1, "name": "Suzie", "user_id": "7qCKAnbiuVieoUnCjotle", "friends": [], "fans": 0,
```

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

```
public static void countTips() {
    try (MongoClient client = new MongoClient( host: "localhost", port: 27017)) {

        MongoDBDatabase database = client.getDatabase( databaseName: "Paweł0142KocimskiA");
        MongoCollection<Document> collection = (MongoCollection<Document>) database.getCollection( "tip");

        Consumer<Document> processBlock = new Consumer<Document>() {
            @Override
            public void accept(Document document) {
                System.out.println(document);
            }
        };

        List<? extends Bson> pipeline = Arrays.asList(
            new Document()
                .append("$match", new Document()
                    .append("date", new Document()
                        .append("$gte", "2012-01-01")
                        .append("$lte", "2012-12-31"))
                ),
            new Document()
                .append("$group", new Document()
                    .append("_id", "$business_id")
                    .append("count", new Document()
                        .append("$sum", 1.0)
                    )
                ),
            new Document()
                .append("$sort", new Document()
                    .append("count", 1.0)
                )
        );

        collection.aggregate(pipeline)
            .allowDiskUse(false)
            .forEach(processBlock);

    } catch (MongoException e) {
        // handle MongoDB exception
    }
}
```

```

Document{{_id=CbUkl9BpdI_m7Yt4yTca_Q, count=1.0}}
Document{{_id=ToSm1CIH3xgaedt63f3FCQ, count=1.0}}
Document{{_id=uTvF-raBw4J9kp_C4jTSYA, count=1.0}}
Document{{_id=FHJQZ0p5ByTHpXUE00B6Uw, count=1.0}}
Document{{_id=MHUX3HRj-gtHlg-YTbIVzQ, count=1.0}}
Document{{_id=fScnL-o3j2G9gUKHa3BaLg, count=1.0}}
Document{{_id=hwJPXF0e0hLXJLlncnUU9Q, count=1.0}}
Document{{_id=joU_S04RAHY4N3lB1TYkKw, count=1.0}}
Document{{_id=DJg1lo8uRmAYJMTmzFv0Vg, count=1.0}}
Document{{_id=Vz-PukBDv5j1UD0YbMbb1w, count=1.0}}
Document{{_id=wrhcXt_x0DSrK6nfcWzBpg, count=1.0}}
Document{{_id=6ySv0bydiT0kP5y1YQImA, count=1.0}}
Document{{_id=5cA0fBYsNagnDot1AATyKA, count=1.0}}
Document{{_id=aTswScd3V3HU0If0IE1Mvw, count=1.0}}
Document{{_id=MLRBLmH9JwUMnJ8yvTpc0Q, count=1.0}}
Document{{_id=s2q0AgLrSnRbfSZU0_zIBA, count=1.0}}
Document{{_id=xzEhmFvpU_MlMnTh2egg, count=1.0}}
Document{{_id=G90xX0E76dEIq51x4RxtVQ, count=1.0}}
Document{{_id=qqWy0Q1KMkFfKp_veyRAfg, count=1.0}}
Document{{_id=P9YDZDqUHVei9_gtRHxBhA, count=1.0}}
Document{{_id=zXR9WUVpo-PF8xH5SmQ_xA, count=1.0}}
Document{{_id=swUzVAsztWfd1W4aZIw61A, count=1.0}}
Document{{_id=rb8rRrH9bcb123-2Qaw3_Q, count=1.0}}
Document{{_id=I7RzNCWg_uVzUJhm9LrB4g, count=1.0}}
Document{{_id=6T0XTVxm9rzx5pJXx5euyQ, count=1.0}}
Document{{_id=vJW16qm0m_U4b30FCa4-yw, count=1.0}}
Document{{_id=SxwV78pKazyCXLuzrkCqUg, count=1.0}}
Document{{_id=2JH55aLSfKvg_711rGcZLQ, count=1.0}}
Document{{_id=rLuuxDPHiP9Z4qtN066eTw, count=1.0}}

```

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

```

public static void avgStars() {
    try (MongoClient client = new MongoClient( host: "localhost", port: 27017)) {

        MongoDB database = client.getDatabase( dbName: "Pawe\u0142KocinskiA");
        MongoCollection<Document> collection = database.getCollection( s: "review");

        Consumer<Document> processBlock = new Consumer<Document>() {
            @Override
            public void accept(Document document) {
                System.out.println(document);
            }
        };

        List<? extends Bson> pipeline = Arrays.asList(
            new Document()
                .append("$group", new Document()
                    .append("_id", "$business_id")
                    .append("avg_stars", new Document()
                        .append("$avg", "$stars")
                    )
                ),
            new Document()
                .append("$match", new Document()
                    .append("avg_stars", new Document()
                        .append("$gte", 4.0)
                    )
                )
        );

        collection.aggregate(pipeline)
            .allowDiskUse(false)
            .forEach(processBlock);
    } catch (MongoException e) {
        // handle MongoDB exception
    }
}

```

```

Document{{_id=BVxLrYWgm1-8TPGMe6CTpg, avg_stars=4.0}}
Document{{_id=BMjggIgOgh8MEXPo8q7q3w, avg_stars=5.0}}
Document{{_id=iyKyJoD0cbK6-MCgvYfMHxw, avg_stars=5.0}}
Document{{_id=yZXEELx18NMwzXCHP345GQ, avg_stars=4.666666666666667}}
Document{{_id=UH48ReVAqpMVjyVroEJAga, avg_stars=4.0}}
Document{{_id=qZk6s0qxtUTrjWw_0je-g, avg_stars=4.333333333333333}}
Document{{_id=0Nwda6h7mRh1QBbx10xhhw, avg_stars=5.0}}
Document{{_id=jTue25w01M0ZtP9Ba718VQ, avg_stars=5.0}}
Document{{_id=0EXVIF07vw-QTTy9V7-ebg, avg_stars=4.75}}
Document{{_id=p0z8fQ1Y8FP3C4ARE9TG_w, avg_stars=5.0}}
Document{{_id=uV2EhAuAmPDxZ0LIgAGFxw, avg_stars=4.666666666666667}}
Document{{_id=NyHw4bpkSLTDB13G2V8bbQ, avg_stars=5.0}}
Document{{_id=VNfCMuY4vngl_nW5WnZ5Lw, avg_stars=4.0}}
Document{{_id=M3G_xbxxfkupkemCZzXh9g, avg_stars=4.8}}
Document{{_id=dtqf1zN4sW0nFoyaaX_ukg, avg_stars=5.0}}
Document{{_id=B0M3_Y-vtg1FswWPUQecLg, avg_stars=4.681818181818182}}
Document{{_id=EK2U2Q2XaZYvcz0XYNCGEw, avg_stars=4.8}}
Document{{_id=nuHftEWNyMIsa4R40pqlQg, avg_stars=4.5}}
Document{{_id=2dxNi0b3ryiPhktzV2WshQ, avg_stars=4.2}}
Document{{_id=q1QRFM3Rh8K9jPMWNus6hA, avg_stars=4.857142857142857}}
Document{{_id=WVE8P8ymHcpYXPsXryPoNQ, avg_stars=4.6}}
Document{{_id=wd3nrdoEp1kmGcf4NX0z1A, avg_stars=5.0}}
Document{{_id=6c-_kLCWVUwnFLXSRUzaSA, avg_stars=4.125}}
Document{{_id=3S_0zqFK_H_68pH9vr0w-Q, avg_stars=5.0}}
Document{{_id=Sc19PSAXk7zRD01aUvCcFw, avg_stars=5.0}}
Document{{_id=GVmnPxW2Xsp02Hbm0HLxTQ, avg_stars=5.0}}
Document{{_id=6_3QPRIym6QqW40TVvXmg, avg_stars=4.857142857142857}}
Document{{_id=_bMEHo_xlg6_vj1njT4YTQ, avg_stars=5.0}}
Document{{_id=e8HQNVQ0cfsTavb0sILCd6w, avg_stars=4.1}}
Document{{_id=XkEqbX22FSKNC1GqrM2tmQ, avg_stars=5.0}}
Document{{_id=9fr-YwF23QpcTjt10Ikc5g, avg_stars=5.0}}
Document{{_id=jbt8LH7tp_m01L2Psj937w, avg_stars=5.0}}
Document{{_id=xiGt6SF0e1S6mHjLD8bK_A, avg_stars=4.25}}
Document{{_id=U4JY0KCgwoD0J1Rk4VuRuQ, avg_stars=4.388888888888889}}
Document{{_id=1N0HhYwuFSBhGDDqUPK1tQ, avg_stars=4.657142857142857}}
Document{{_id=uUP58_KLQEuvgZfrVaWfRRQ, avg_stars=5.0}}
Document{{_id=snH-0mYyq3MbFKsu6mpRdA, avg_stars=4.666666666666667}}
Document{{_id=rR6CJ1eHbXT6D1D9If_dag, avg_stars=4.875}}
Document{{_id=CgY6echEKYViP0kg2D3rJg, avg_stars=4.841666666666667}}
Document{{_id=vLA9NDe04DNnFG31CnEQTA, avg_stars=4.090909090909091}}

```

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

```

private static void deleteBusinessStar()
{
    db.getCollection( name: "business").deleteMany(eq( fieldName: "stars", value: 2));
}

```

- 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

Baza Sklep, którą chcę zaproponować będzie składała się z 3 tabel

1) Klienci - informacje o klientach i ich zakupach

```
1 use Sklep
2 function addClient(imie, nazwisko, pesel, adres)
3 {
4     db.Klienci.insert({
5         imie: imie,
6         nazwisko: nazwisko,
7         pesel: pesel,
8         adres: adres,
9         zakupy: []
10    })
11 }
12
13 addClient("Paweł", "Kocimski", "123456789", "Kraków ul.nieznana 1");
14 addClient("Jan", "Kowalski", "09856789", "Warszawa ul.pólna 2");
15 addClient("Michał", "Nowak", "123456789", "Gdańsk ul. szybka 3");
16
```

Baza danych:

```
1 {
2   "_id" : ObjectId("5e961ebc46b8527b857bcc7a"),
3   "imie" : "Paweł",
4   "nazwisko" : "Kocimski",
5   "pesel" : "123456789",
6   "adres" : "Kraków ul.nieznana 1",
7   "zakupy" : [
8   ]
9 }
10
11 {
12   "_id" : ObjectId("5e961ebc46b8527b857bcc7b"),
13   "imie" : "Jan",
14   "nazwisko" : "Kowalski",
15   "pesel" : "09856789",
16   "adres" : "Warszawa ul.pólna 2",
17   "zakupy" : [
18   ]
19 }
20
21 {
22   "_id" : ObjectId("5e961ebc46b8527b857bcc7c"),
23   "imie" : "Michał",
24   "nazwisko" : "Nowak",
25   "pesel" : "123456789",
26   "adres" : "Gdańsk ul. szybka 3",
27   "zakupy" : [
28   ]
29 }
30 }
```

2) Zakupy - zakupione towary przez klienta, zbiór produktów

```

1 use Sklep
2 function addShopping(data, dostawa, rabat, produkty)
3 {
4     db.Zakupy.insert({
5         data: data,
6         dostawa: dostawa,
7         rabat: rabat,
8         produkty: []
9     })
10 }
11 addShopping(new Date("2016-05-18T16:00:00Z"), true, 0.2)
12 addShopping(new Date("2019-07-18T16:00:00Z"), false, 0)
13 addShopping(new Date("2020-02-11T16:00:00Z"), true, 0.1)

```

Baza Danych:

```

1 {
2     "_id" : ObjectId("5e9627fa46b8527b857bcc7d"),
3     "data" : ISODate("2016-05-18T16:00:00.000+0000"),
4     "dostawa" : true,
5     "rabat" : 0.2,
6     "produkty" : [
7
8     ]
9 }
10 {
11     "_id" : ObjectId("5e9627fa46b8527b857bcc7e"),
12     "data" : ISODate("2019-07-18T16:00:00.000+0000"),
13     "dostawa" : false,
14     "rabat" : 0.0,
15     "produkty" : [
16
17     ]
18 }
19 {
20     "_id" : ObjectId("5e9627fa46b8527b857bcc7f"),
21     "data" : ISODate("2020-02-11T16:00:00.000+0000"),
22     "dostawa" : true,
23     "rabat" : 0.1,
24     "produkty" : [
25
26     ]
27 }
28

```

3) Produkty – przedmioty, które są w sklepie

```

1 use Sklep
2 function addProdukt(nazwa, kategoria, cena, przecena, gwarancja, opis)
3 {
4     db.Produkty.insert({
5         nazwa: nazwa,
6         kategoria: kategoria,
7         cena: cena,
8         przecena: przecena,
9         gwarancja: gwarancja,
10        opis: opis
11    })
12 }
13 addProdukt("kalosze", "odzież", 59, 0, "2 lata", "firmowe")
14 addProdukt("masło", "nabiał", 1.99, 0.1, false, "z okregowej mleczarni")
15 addProdukt("gazeta", "prasa", 1.99, 0, false, "dziennik prawicowy")

```

Baza danych:

```

1 {
2   "_id" : ObjectId("5e962ae6ca3ecc66f0f4030f"),
3   "nazwa" : "kalosze",
4   "kategoria" : "odzież",
5   "cena" : 59.0,
6   "przecena" : 0.0,
7   "gwarancja" : "2 lata",
8   "opis" : "firmowe"
9 }
10 {
11   "_id" : ObjectId("5e962ae6ca3ecc66f0f40310"),
12   "nazwa" : "masło",
13   "kategoria" : "nabiał",
14   "cena" : 1.99,
15   "przecena" : 0.1,
16   "gwarancja" : false,
17   "opis" : "z okregowej mleczarni"
18 }
19 {
20   "_id" : ObjectId("5e962ae6ca3ecc66f0f40311"),
21   "nazwa" : "gazeta",
22   "kategoria" : "prasa",
23   "cena" : 1.99,
24   "przecena" : 0.0,
25   "gwarancja" : false,
26   "opis" : "dziennik prawicowy"
27 }
28

```