

Bazy danych – NoSQL MongoDB – zadania

Imię i nazwisko: Krzysztof Hardek

Tydzień A/B, godz lab.: Wtorek A, 9:35

Zadania:

1. Wykorzystując bazę danych **yelp dataset** 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.

```
db.getCollection('business').aggregate([  
  
  {  
  
    $group: { _id: "$city" }  
  
  },  
  
  {  
  
    $sort: { _id: 1 }  
  
  }  
  
])
```

```
/* 1 */  
{  
  "_id" : "Ahwatukee"  
}  
  
/* 2 */  
{  
  "_id" : "Anthem"  
}  
  
/* 3 */  
{  
  "_id" : "Apache Junction"  
}  
  
/* 4 */  
{  
  "_id" : "Arcadia"  
}
```

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

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

880318

- 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})
```

```
/* 1 */
{
  "_id" : ObjectId("5e85cf7d7a7d96a76ab2e96e"),
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
  "name" : "Charter Communications",
  "stars" : 1.5
}

/* 2 */
{
  "_id" : ObjectId("5e85cf7d7a7d96a76ab2e979"),
  "full_address" : "6401 University Ave\nMiddleton, WI 53562",
  "name" : "Crandalls Carryout & Catering",
  "stars" : 4.0
}

/* 3 */
{
  "_id" : ObjectId("5e85cf7d7a7d96a76ab2e981"),
  "full_address" : "6230 University Ave\nMiddleton, WI 53562",
  "name" : "Mi Cocina",
  "stars" : 3.0
}

/* 4 */
{
  "_id" : ObjectId("5e85cf7d7a7d96a76ab2e9a2"),
  "full_address" : "6625 Century Ave\nMiddleton, WI 53562",
  "name" : "Stamm House At Pheasant Branch",
  "stars" : 2.0
}

/* 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.

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

```
/* 1 */
{
  "_id" : ObjectId("5e85d793c20e54a0d0424473"),
  "yelping_since" : "2014-04",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 3,
  "name" : "A.J.",
  "user_id" : "eWUOSJ5I5MiBMT5T3vJo5A",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}

/* 2 */
{
  "_id" : ObjectId("5e85d793c20e54a0d0424795"),
  "yelping_since" : "2013-04",
  "votes" : {
    "funny" : 0,
    "useful" : 0,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "A.J.",
  "user_id" : "ZhptqExMepJ0vnbrosJwAg",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 5.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}
```

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

```
db.getCollection('tip').aggregate([
  {
    $match: {date: {$regex: "2012-*"}}
  },
  {
```

```

    $group: {_id: "$business_id", total: { $sum: 1 }}
  },
  {
    $sort: { total: 1 }
  },
  {
    $lookup: {
      from: "business",
      localField: "_id",
      foreignField: "business_id",
      as: "business"
    }
  },
  {
    $project: {"business.name": 1, total: 1}
  }
]
})

```

```

/* 1 */
{
  "_id" : "rrfmaxtcmg0h26Vyg6Eebg",
  "total" : 1.0,
  "business" : [
    {
      "name" : "Poisoned Pen A Mystery Book Store"
    }
  ]
}

/* 2 */
{
  "_id" : "lT0ojiymSxa_hqG5i9lUmg",
  "total" : 1.0,
  "business" : [
    {
      "name" : "Samba Brazilian Grill"
    }
  ]
}

```

- f. Wyznacz, jaka 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([
  {
    $match: { stars: {$gte: 4}}
  },
  {
    $group: {_id: "$business_id", avg: {$avg: "$stars"}}
  },
  {
    $lookup: {

```

```

        from: "business",
        localField: "_id",
        foreignField: "business_id",
        as: "b"
    }
},
{
    $unwind: { path: "$b" }
},
{
    $project: { "b.name": 1, avg: 1}
}
])

```

```

/* 1 */
{
  "_id" : "lo2tx5L7cwNNLUb42qbtTw",
  "avg" : 4.333333333333333,
  "b" : {
    "name" : "Al Dente"
  }
}

/* 2 */
{
  "_id" : "MWFKSDW4cLw20RRjkKJTcA",
  "avg" : 4.333333333333333,
  "b" : {
    "name" : "Obi Nail Salon"
  }
}

/* 3 */
{
  "_id" : "3NQ8_b-aQw9tpgglyytfYg",
  "avg" : 4.75,
  "b" : {

```

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

```
db.getCollection('business').remove({stars: 2})
```

```
Removed 1576 record(s) in 71ms
```

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

```

function insertReview(user_id, stars, text, business_id){

    db.review.insert({

```

```

        user_id: user_id,

        stars: stars,

        date: Date(),

        text: text,

        business_id: business_id,

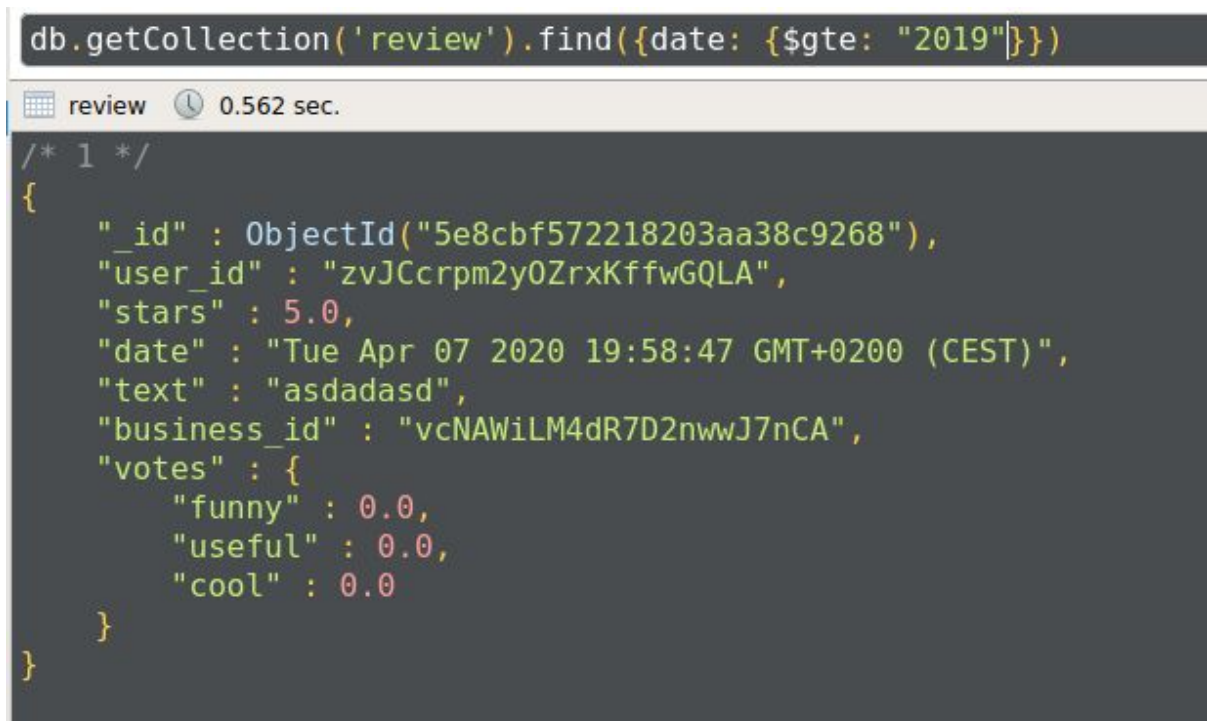
        votes: { funny: 0 ,useful: 0,cool: 0 }

    });

}

insertReview("zvJCcrpm2y0ZrxKffwGQLA", 5, "asdadasd", "vcNAWiLM4dR7D2nwwJ7nCA")

```



The screenshot shows a MongoDB console interface. At the top, a query is entered: `db.getCollection('review').find({date: {$gte: "2019"}})`. Below the query, the console shows the execution time as 0.562 sec. The result is a single document, indicated by `/* 1 */`. The document is a JSON object with the following fields: `_id` (ObjectId), `user_id` (zvJCcrpm2y0ZrxKffwGQLA), `stars` (5.0), `date` (Tue Apr 07 2020 19:58:47 GMT+0200 (CEST)), `text` (asdadasd), `business_id` (vcNAWiLM4dR7D2nwwJ7nCA), and `votes` (an object with funny: 0.0, useful: 0.0, and cool: 0.0).

```

db.getCollection('review').find({date: {$gte: "2019"}})

review 0.562 sec.

/* 1 */
{
  "_id" : ObjectId("5e8cbf572218203aa38c9268"),
  "user_id" : "zvJCcrpm2y0ZrxKffwGQLA",
  "stars" : 5.0,
  "date" : "Tue Apr 07 2020 19:58:47 GMT+0200 (CEST)",
  "text" : "asdadasd",
  "business_id" : "vcNAWiLM4dR7D2nwwJ7nCA",
  "votes" : {
    "funny" : 0.0,
    "useful" : 0.0,
    "cool" : 0.0
  }
}

```

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.

```

function getCategoryBusiness(category){
    return db.business.find( {categories: {$all: [category]} })
}

getCategoryBusiness("Mass Media")

```



```

/* 1 */
{
  "_id" : ObjectId("5e85cf7d7a7d96a76ab2e96e"),
  "business_id" : "oLctHIA1Axmsg0uu4dM6Vw",
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",
  "hours" : {},
  "open" : false,
  "categories" : [
    "Television Stations",
    "Mass Media"
  ],
  "city" : "Mc Farland",
  "review_count" : 10,
  "name" : "Charter Communications",
  "neighborhoods" : [],
  "longitude" : -89.3229199,
  "state" : "WI",
  "stars" : 1.5,
  "latitude" : 42.9685074,
  "attributes" : {},
  "type" : "business"
}

/* 2 */
{
  "_id" : ObjectId("5e85cf7d7a7d96a76ab2eaf3"),

```

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

```

function changeUserName(name, id){
  db.user.update({user_id: id}, {$set: {name: name}})
}

changeUserName("krzys", "5Xh4Qc3rxhAQ_NcNtxLssQ")

```

```
db.getCollection('user').find({user_id: "5Xh4Qc3rxhAQ_NcNtxLssQ"})
```

user 0.12 sec.

```
/* 1 */
{
  "_id" : ObjectId("5e85d78bc20e54a0d04041a1"),
  "yelping_since" : "2012-01",
  "votes" : {
    "funny" : 0,
    "useful" : 1,
    "cool" : 0
  },
  "review_count" : 1,
  "name" : "krzys",
  "user_id" : "5Xh4Qc3rxhAQ_NcNtxLssQ",
  "friends" : [],
  "fans" : 0,
  "average_stars" : 1.0,
  "type" : "user",
  "compliments" : {},
  "elite" : []
}
```

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

```
var mapF = function(){
  emit(this.business_id, 1);
};

var reduceF = function(id, tip_count){
  return Array.sum(tip_count)
};

db.tip.mapReduce(
  mapF,
  reduceF,
  {out: "tip_count"}
)
```



```
db.getCollection("tip_count").find({})
```

tip_count 0.001 sec.

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

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

/* 3 */
{
  "_id" : "--BlvD0_RG2yElKu9XA1_g",
  "value" : 21.0
}

/* 4 */
{
  "_id" : "--Dl2rW_x08GuYBomlg9zw",
  "value" : 2.0
}

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

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.

```
public AggregateIterable<Document> getAllCities(){

    Document group = new Document("$group", new Document("_id", "$city"));

    Document sort = new Document("$sort", new Document("_id", 1));

    MongoCollection<Document> collection = db.getCollection("business");
```

```
List<Document> pipeline = Arrays.asList(group, sort);

return collection.aggregate(pipeline);
}
```

```
Document{{_id=Ahwatukee}}
Document{{_id=Anthem}}
Document{{_id=Apache Junction}}
Document{{_id=Arcadia}}
Document{{_id=Atlanta}}
Document{{_id=Avondale}}
Document{{_id=Black Canyon City}}
Document{{_id=Bonnyrigg}}
Document{{_id=Boulder City}}
Document{{_id=Buckeye}}
```

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

```
public long getReviewsAfter2011(){
    MongoCollection<Document> collection = db.getCollection("review");
    var query = new Document("date", new BasicDBObject("$gte", "2011"));
    var docs = new ArrayList<Document>();

    return collection.countDocuments(query);
}
```

880319

// inny wynik niż w zad 1 z racji takiej że dodałem jedną recenzję.

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

```
public FindIterable<Document> getClosedBusiness(){
    MongoCollection<Document> collection = db.getCollection("business");
    return collection.find(eq("open", false)).projection(include("name",
"full_address", "stars"));
}
```

```
Document{{_id=5e85cf7d7a7d96a76ab2e96e, full_address=4156 County Rd B
Mc Farland, WI 53558, name=Charter Communications, stars=1.5}}
Document{{_id=5e85cf7d7a7d96a76ab2e979, full_address=6401 University Ave
Middleton, WI 53562, name=Crandalls Carryout & Catering, stars=4.0}}
Document{{_id=5e85cf7d7a7d96a76ab2e981, full_address=6230 University Ave
Middleton, WI 53562, name=Mt Cocina, stars=3.0}}
Document{{_id=5e85cf7d7a7d96a76ab2e9b2, full_address=1901 Cayuga St
Middleton, WI 53562, name=Soup Factory, stars=3.0}}
Document{{_id=5e85cf7d7a7d96a76ab2e9c1, full_address=1632 W Main St
Sun Prairie, WI 53590, name=Deli Roma, stars=4.0}}
```

- 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 FindIterable<Document> getBadUsers(){
    MongoCollection<Document> collection = db.getCollection("user");

    return collection.find(and(
        eq("votes.funny", 0),
        eq("votes.useful", 0)
    )).projection(include("name")).sort(ascending("name"));
}

// Dokonałem dodatkowej projekcji, aby wynik zmieścił się w dokumencie oraz
// był czytelny.
```

```
Document({_id=5e85d794c20e54a0d0426eca, name=Joseph})
Document({_id=5e85d794c20e54a0d0428729, name=Joseph})
Document({_id=5e85d794c20e54a0d0429f50, name=Joseph})
Document({_id=5e85d795c20e54a0d042cb9a, name=Joseph})
Document({_id=5e85d796c20e54a0d0431354, name=Joseph})
Document({_id=5e85d796c20e54a0d043209a, name=Joseph})
Document({_id=5e85d796c20e54a0d04328e9, name=Joseph})
Document({_id=5e85d796c20e54a0d0432a2e, name=Joseph})
Document({_id=5e85d797c20e54a0d0434970, name=Joseph})
Document({_id=5e85d797c20e54a0d0434ab1, name=Joseph})
Document({_id=5e85d797c20e54a0d0434e46, name=Joseph})
Document({_id=5e85d797c20e54a0d0435868, name=Joseph})
Document({_id=5e85d797c20e54a0d0436c5d, name=Joseph})
```

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

```
public AggregateIterable<Document> getBusinessTips() {
    MongoCollection<Document> collection = db.getCollection("tip");
    return
        collection.aggregate(
            Arrays.asList(
                Aggregates.match(regex("date", "2012-*")),
                Aggregates.group("$business_id",
                    Accumulators.sum("total", 1)),
                Aggregates.sort(ascending("total")),
                Aggregates.lookup("business", "_id", "business_id",
                    "b"),
                Aggregates.project(
                    Projections.fields(
                        Projections.include("b.name", "total")
                    )
                )
            )
        );
}
```

```
Document{{_id=km-sxUVPKrmjsoyK6BYGgA, total=1, b=[Document{{name=Edinburgh College Of Art}}]}}
Document{{_id=nJZzBj2sBamL-FSIkrM4eg, total=1, b=[Document{{name=Petco}}]}}
Document{{_id=38n7vTuYBUVunJN9IAcMmw, total=1, b=[Document{{name=OfficeMax}}]}}
Document{{_id=moL0xxSeGhMXzPt2GH_i3Q, total=1, b=[Document{{name=Walmart}}]}}
Document{{_id=Y37LBcBEni--WrFsK-018g, total=1, b=[Document{{name=The Walking Company}}]}}
Document{{_id=B0UAGNPffEYg2-cXSTBeRA, total=1, b=[Document{{name=Collision Authority}}]}}
Document{{_id=6A_m3LgSfDutiBwdAwybaA, total=1, b=[Document{{name=Coronado Bay Club Apartments}}]}}
Document{{_id=c0XWFX-St7IvaLADHK8c6A, total=1, b=[Document{{name=Pet Club}}]}}
Document{{_id=CBumPtpjISns8paiSwTLw, total=1, b=[Document{{name=Stage Deli}}]}}
Document{{_id=rz1k1xr5G9Flm0QqzWmxBA, total=1, b=[Document{{name=Papago Park}}]}}
Document{{_id=3mKAYa9iG-ej5ow6_NtVnA, total=1, b=[Document{{name=Great Clips}}]}}
Document{{_id=sfax6iIV4DYMWhCJshlXIQ, total=1, b=[Document{{name=Starbucks - Safeway}}]}}
```

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

```
public AggregateIterable<Document> getAvgBusinessStars() {
    MongoCollection<Document> collection = db.getCollection("review");
    return
        collection.aggregate(
            Arrays.asList(
                Aggregates.match(gte("stars", 4)),
                Aggregates.group("$business_id", Accumulators.avg("avg",
"$stars")),
                Aggregates.lookup("business", "_id", "business_id",
"b"),
                Aggregates.unwind("$b"),
                Aggregates.project(
                    Projections.fields(
                        Projections.include("b.name", "avg")
                    )
                )
            )
        );
}
```

```
Document{{_id=p6eUvL6rynqzLrB6d2iQ-Q, avg=5.0, b=Document{{name=Perfecta LMT}}}}
Document{{_id=XHr5mXFgob0HoxbPJxmYdg, avg=4.451127819548872, b=Document{{name=Scramble A Breakfast Joint}}}}
Document{{_id=VQs48-mgR4bXCzL6I8Nb5g, avg=4.333333333333333, b=Document{{name=Aspen Dental}}}}
Document{{_id=Sw0pDQKzToVJNL3i_tJf6Q, avg=5.0, b=Document{{name=1st Glass Window Cleaners, LLC}}}}
Document{{_id=pNJhovUsGAZ4Xq63LCU-5g, avg=4.25, b=Document{{name=Einstein Bros Bagels}}}}
Document{{_id=dNeIr3C0NcStusZSs38Pmg, avg=4.0, b=Document{{name=Wild Horse Cafe}}}}
Document{{_id=bUQC1_Z3TZBoz46EqXko4w, avg=5.0, b=Document{{name=Circle K Stores}}}}
Document{{_id=u09jlgjrV3_Td4hj_WfDSg, avg=4.375, b=Document{{name=Babies R US the Baby Superstore}}}}
Document{{_id=ihBjJvt1m03V4WkcB0oHHA, avg=4.5, b=Document{{name=Big Stitch Embroidery}}}}
Document{{_id=tcn6SVQRyEu3aa2KktRR9g, avg=4.545454545454546, b=Document{{name=Bisbee Breakfast Club Mesa}}}}
Document{{_id=n2vuEfMHtM1ZE2LR9ppLXg, avg=4.368421052631579, b=Document{{name=Jacky Chan}}}}
Document{{_id=-X128ZdT1oEka2AYsFN_KA, avg=5.0, b=Document{{name=Eso}}}}
```

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

```
public DeleteResult removeBadBusiness() {

    MongoCollection<Document> collection = db.getCollection("business");

    return collection.deleteMany(eq("stars", 2));

}
```

```
AcknowledgedDeleteResult{deletedCount=0}
```

// Jest zero ponieważ już wcześniej usunąłem wszystkie takie firmy.

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

Stworzenie bazy:

use Company

```
> use Company  
switched to db Company
```

Dodanie kolekcji:

```
db.clients.insert({ name: "Krzysztof", surname: "Hardek", address: { city: "Oswiecim",  
street: "Bema", zip_code: "32-600" }})
```

```
WriteResult({ "nInserted" : 1 })
```

```
db.purchases.insert({client_id: ObjectId("5e90a152f71dd394eb9b9d41"), date:"2020-04-10",  
item_quantity: 2, value: 100, categories: ["Food", "Alcohol"]})
```

```
Inserted 1 record(s) in 19ms
```

```
db.purchase_item.insert({purchase_id: ObjectId("5e90a2aebaf88bb7b9e4bb36"), name:"Apple",  
category: "Food", price: 50})
```

```
Inserted 1 record(s) in 15ms
```

Po dodaniu pozostałych dokumentów struktura bazy wygląda następująco:

Klienci:

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

clients 0.001 sec.

```
/* 1 */
{
  "_id" : ObjectId("5e90a152f71dd394eb9b9d41"),
  "name" : "Krzysztof",
  "surname" : "Hardek",
  "address" : {
    "city" : "Oswiecim",
    "street" : "Bema",
    "zip_code" : "32-600"
  }
},
/* 2 */
{
  "_id" : ObjectId("5e90a757baf88bb7b9e4bb39"),
  "name" : "Dawid",
  "surname" : "Sliwinski",
  "address" : {
    "city" : "Oswiecim",
    "street" : "Zaborska",
    "zip_code" : "32-600"
  }
}
```


Zakupy:

```
/* 1 */
{
  "_id" : ObjectId("5e90a2aebaf88bb7b9e4bb36"),
  "client_id" : ObjectId("5e90a152f71dd394eb9b9d41"),
  "date" : "2020-04-10",
  "item_quantity" : 2.0,
  "value" : 100.0,
  "categories" : [
    "Food",
    "Alcohol"
  ]
}

/* 2 */
{
  "_id" : ObjectId("5e90a7dfbaf88bb7b9e4bb3a"),
  "client_id" : ObjectId("5e90a757baf88bb7b9e4bb39"),
  "date" : "2020-04-9",
  "item_quantity" : 1.0,
  "value" : 100.0,
  "categories" : [
    "Music"
  ]
}
```


Przedmioty zakupu:

```
/* 1 */
{
  "_id" : ObjectId("5e90a396baf88bb7b9e4bb37"),
  "purchase_id" : ObjectId("5e90a2aebaf88bb7b9e4bb36"),
  "name" : "Apple",
  "category" : "Food",
  "price" : 50.0
}

/* 2 */
{
  "_id" : ObjectId("5e90a3d2baf88bb7b9e4bb38"),
  "purchase_id" : ObjectId("5e90a2aebaf88bb7b9e4bb36"),
  "name" : "Beer",
  "category" : "Alcohol",
  "price" : 50.0
}

/* 3 */
{
  "_id" : ObjectId("5e90a81dbaf88bb7b9e4bb3b"),
  "purchase_id" : ObjectId("5e90a757baf88bb7b9e4bb39"),
  "name" : "The Doors",
  "category" : "Music",
  "price" : 100.0
}
```

Uzasadnienie:

Dokumenty w kolekcji klienci zawierają informacje potrzebne do zidentyfikowania klienta. Pole address jest dokumentem JSON ze względu na kompleksową naturę obiektu jakim jest adres zamieszkania. Zakupy stanowią kolekcje dokumentów reprezentujących przedmioty kupione za jednym razem (może ich być kilka). Pole categories jest tablicą i reprezentuje zbiór kategorii do jakich należą przedmioty kupione podczas tego zakupu. Długość tej tablicy może być różna. Kolekcja purchase_item przechowuje przedmioty możliwe do kupienia. Składają się one wszystkie na zakupy.