Wydział WIMilP	Imię i nazwisko 1.Kacper Bielak	Rok 2	Grupa GL01			
Temat:						
Zadanie nr.1 - MongoDB						
Data wykonania		Przedmiot:		OCENA		
28.05.2021	Bazy danych					

Podstawy MongoDB

1. Utworzenie nowej bazy danych oraz kolekcji students o zadanych atrybutach w zadaniu i wstawienie kilku rekordów:

```
Kod:
use nowa_baza
db.createCollection('students')
db.myData.insert({ no album: '400000', name: 'Kacper', surname: 'Bielak',
birth_date: '20.10.2000', pesel: '12345678912', address: 'Dzwola 1', phone_number:
'123123123', mail_address: 'xxxxx@o2.pl', field_of_study: 'IT', faculty: 'WIMiIP'}
db.myData.insert({ no_album: '400011', name: 'Mateusz', surname: 'Gryn',
birth_date: '20.09.2000', pesel: '12345421443', address: 'Dzwola 5', phone_number:
'11111111', mail_address: 'yyy@o2.pl', field_of_study: 'IT', faculty: 'WIMiIP'})
db.myData.insert({ no_album: '400012', name: 'Lukasz', surname: 'Gra', birth_date:
'01.07.2000', pesel: '12345421455', address: 'Krakow 5', phone_number:
'22222222', mail_address: 'aba@o2.pl', field_of_study: 'IO', faculty: 'WIMiIR'})
db.myData.insert({ no_album: '400013', name: 'Maja', surname: 'Grak', birth_date:
'01.05.2000', pesel: '12345421400', address: 'Krakow 10', phone_number:
'33333333', mail_address: 'eee@o2.pl', field_of_study: 'XX', faculty: 'WIMiIR'})
db.myData.insert({ no_album: '400014', name: 'Bartek', surname: 'Dab', birth_date:
'01.03.2000', pesel: '12345421451', address: 'Krakow 102', phone_number:
 '333334443', mail_address: 'yeye@o2.pl', field_of_study: 'LOL', faculty: 'EITI'})
Zrzut ekranu:
  db.createCollection('myData')
  "ok': 1 }
db.myOata.find()
db.myOata.find()
db.myOata.insert({
...o.albus: '400000', name: 'Kacper', surname: 'Bielak', birth_date: '20.10.2000', pesel: '12345678912', address: 'Dzwola 1', phone_number: '123123123', mail_address: 'xxxxxx(e)c.pl',
...field_of_study: 'IT', faculty: 'WIMIP'))
ncaught exception: ReferenceError: dm is not defined :
     .myOata.insert({ no_album: '400000', name: 'Kacper', surname: 'Bielak', birth_date: '20.10.2000', pesel: '12345678912', address: 'Dzwola 1', phone_number: '123123123', mail_address: 'xxxxxx@02.pl', field_of_
     - (Nousse, Manual ))
- Result(| Inlinerted : 1 )
- Modern inserted : 1 )
- Mod
```

```
talinsert(( no album: '400011', name: 'Mateusz', surname: 'Gryn', birth_date: '20.09.2000', pesel: '12345421443', address: 'Dzwola 5', phone_number: '111111111', mail_address: 'yyy@02.pl', field_of_stud
   Result({ "nInserted" : 1 })
myData.insert([ no_album: '400012', name: 'Lukasz', surname: 'Gra', birth_date: '01.07.2000', pesel: '12345421455', address: 'Krakow 5', phone_number: '222222222', mail_address: 'aba@o2.pl', field_of_study
   nesult(["ninested": 1])
myOata.insert((no_album: '400013', name: 'Maja', surname: 'Grak', birth_date: '01.05.2000', pesel: '12345421400', address: 'Krakow 10', phone_number: '33333333', mail_address: 'eee@o2.pl', field_of_study
myOata.insert((no_album: '400013', name: 'Maja', surname: 'Grak', birth_date: '01.05.2000', pesel: '12345421400', address: 'Krakow 10', phone_number: '333333333', mail_address: 'eee@o2.pl', field_of_study
telesult("minserted": 1 })
b.myOata.insert("no.album: '400014', name: 'Bartek', surname: 'Dab', birth_date: '01.03.2000', pesel: '12345421451', address: 'Krakow 102', phone_number: '333334443', mail_address: 'yeye@o2.pl', field_of_stu
'\O4.' faculty: 'EITI')
telesult("minserted": 1 })
```

```
no.myolara.ind() [.dd : ObjectId("6601071d556d9e18a00f81af"), "no_album" : "400000", "name" : "Kacper", "surname" : "Bielak", "birth_date" : "20.10.2000", "pesel" : "12345678912", "address" : "Dzwola 1", "phone_number" : "12: "1234578912", "field_of_study" : "IT", "faculty" : "MINIP" )
[.dd : ObjectId("6001074655609e18a00f81200"), "no_album" : "4000011", "name" : "Mateusz", "surname" : "Gryn", "birth_date" : "20.10.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone_number" : "1111
[.] "nail_address : ".yygob_p.l", "field_of_study" : "IT", "faculty" : "MINIP!" )
[.] "d : ObjectId("60010820656509e18a00f81b1"), "no_album" : "400012", "name" : "Lukasz", "surname" : "Gra", "birth_date" : "01.07.2000", "pesel" : "12345421455", "address" : "Krakow 5", "phone_number" : "222222
                                                                                                                                                                                                                                                                                                                                                                                                                                       "Mateusz", "surname" : "Gryn", "birth date" : "20.09.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone number" : "1111
                              : ObjectId("60010746560Je1800161801), "no album": "400011", "name": "Mateusz", "surname": "Gryn", "birth_date": "20.09.2000", "pesel": "12345421443", "address": "Dzwola 5", "phone_number": "1111 "moil_addness": "Nyy@o.pl", "field_of_study": "II", "faculty": "NINIIP" |
: ObjectId("6001082065650Je1800461011), "no album": "400012", "name": "Lukasz", "surname": "Gra", "birth_date": "01.07.2000", "pesel": "12345421455", "address": "Krakow 5", "phone_number": "222222 ail_address": "aba@o.pl", "field_of_study": "IO", "faculty": "NINIIR" |
: ObjectId("60010855650Je18004618015"), "no album": "400013", "name": "Maja", "surname": "Grak", "birth_date": "01.05.2000", "pesel": "12345421400", "address": "Krakow 10", "phone_number": "333333 ail_address": "ee@o2.pl", "field_of_study": "XX", "faculty": "NINIIR" |
: ObjectId("6001080e6560Je180016105"), "no album": "400014", "name": "Bartek", "surname": "Dab", "birth_date": "01.03.2000", "pesel": "12345421451", "address": "Krakow 102", "phone_number": "3333 "mail_address": "Yey@o.pl", "field_of_study": "Krakow 102", "phone_number": "3333 "mail_address": "Syey@o.pl", "field_of_study": "Krakow 102", "phone_number": "3333 "mail_address": "Yey@o.pl", "field_of_study": "Krakow 102", "phone_number": "Bartek", "yew@o.pl", "field_of_study": "Krakow 102", "phone_number": "Bartek", "yew@o.pl", "field_of_study": "Krakow 102", "phone_number": "Barte
```

2. Wstawienie rekordów zawierających:

a. Mniej atrybutów:

Kod:

```
db.myData.insert({ no_album: '400014', name: 'Bartek', surname: 'Dab', birth_date: '01.03.2000', pesel: '12345421451', address: 'Krakow 102', phone_number: '333334443'})
```

Zrzut:

```
> db.myData.insert({ no_album: '400014', name: 'Bartek', surname: 'Dab', birth_date: '01.03.2000', pesel: '12345421451', address: 'Krakow 102', phone_number: '333334443'})
WriteResult({ "nInserted" : 1 })
```

Efekt po wypisaniu:

```
{ "_id" : ObjectId("60b10a46656d9e18a0bf81b4"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_dat
" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443" }
```

Wniosek:

Dodając rekord zawierający mniej atrybutów również jest dodany, od pozostałych różni się po prostu inną liczbą atrybutów, niedodane nie istnieją, nie mają również wartości null, po prostu ich nie ma.

b. Więcej atrybutów:

Kod:

```
db.myData.insert({ no_album: '400014', name: 'Bartek', surname: 'Dab', birth_date: '01.03.2000', pesel: '12345421451', address: 'Krakow 102', phone_number: '333334443', mail_address: 'yeye@o2.pl', field_of_study: 'LOL', faculty: 'EITI', dodatkowy1: 'tak', dodatkowy2: 'nie'})
```

Zrzut:

```
> db.myData.insert({ no_album: '400014', name: 'Bartek', surname: 'Dab', birth_date: '01.03.2000', pesel: '12345421451'
address: 'Krakow 102', phone_number: '333334443', mail_address: 'yeye@o2.pl', field_of_study: 'LOL', faculty: 'EITI',
odatkowy1: 'tak', dodatkowy2: 'nie'})
WriteResult({ "nInserted" : 1 })
```

Efekt po wypisaniu:

Wniosek:

Sytuacja podobna jak wcześnie. Rekord został dodany, od pozostałych różni się po prostu inną liczbą atrybutów, jest ich więcej.

3. CRUD

a. Create - dodanie nowego rekordu

Kod

```
db.myData.insert({ no_album: '400000', name: 'Kacper', surname: 'Bielak', birth_date: '20.10.2000', pesel: '12345678912', address: 'Dzwola 1', phone_number: '123123123', mail_address: 'xxxxx@o2.pl', field_of_study: 'IT', faculty: 'WIMiIP'}
```

Zrzut ekranu:

> db.myOdta.insert([no.album: '400000', name: 'Kacper', surname: 'Bielak', birth_date: '20.10.2000', pesel: '12345678912', address: 'Dzwola 1', phone_number: '123123123', mail_address: 'xxxxx@02.pl', field_of_tudy: 'IT', faculty: 'MIMIIP')
WriteResult(["ninserted" : 1])

b. Read - wyświetlenie wszystkich rekordów

Kod:

db.myData.find()

Zrzut ekranu:

```
rzut ekranu:
db.myData.find()
    ".id" : ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "4000000", "name" : "Kacper", "surname" : "Bielak", "birth_d
    e" : "20.10.2000", "pesel" : "12345678912", "address" : "Dzwola 1", "phone_number" : "123123123", "mail_address" : "xx
    xx@02.pl", "field_of_study" : "IT", "faculty" : "MIMIIP" }
    "_id" : ObjectId("60b1071d656d9e18a0bf81b0"), "no_album" : "400011", "name" : "Mateusz", "surname" : "Gryn", "birth_da
    ": "20.09.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone_number" : "1111111111, "mail_address" : "yyy
    22.pl", "field_of_study" : "IT", "faculty" : "MIMIIP" }
    "_id" : ObjectId("60b1082b656d9e18a0bf81b1"), "no_album" : "400012", "name" : "Lukasz", "surname" : "Gra", "birth_date
    "01.07.2000", "pesel" : "12345421455", "address" : "Krakow 5", "phone_number" : "222222222", "mail_address" : "ab@0
    pl,", "field_of_study" : "IO", "faculty" : "MIMIIR" }
    "_id" : ObjectId("60b10855656d9e18a0bf81b2"), "no_album" : "400013", "name" : "Maja", "surname" : "Grak", "birth_date"
    "01.05.2000", "pesel" : "12345421400", "address" : "Krakow 10", "phone_number" : "333333333", "mail_address" : "eee@0
    pl,", "field_of_study" : "XX", "faculty" : "MIMIIR" }
    "_id" : ObjectId("60b1083e656d9e18a0bf81b3"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
    "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443", "mail_address" : "yey
    plo.2,pl", "field_of_study" : "LOL", "faculty" : "EITI" }
    "_id" : ObjectId("60b1084656d9e18a0bf81b4"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
    "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443", "mail_address" : "yey
    plo.2,pl", "field_of_study" : "LOL", "faculty" : "EITI" }
    "_id" : ObjectId("60b108465669e18a0bf81b4"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
    "01.03.2000", "pesel" : "12345421451", "address" : "Kra
```

Update – zmiana numer albumu studenta

Update można wykonać na dwa sposoby:

Rekord zostanie update'owany w taki sposób, że wszystkie informację zostaną nadpisane. W przypadku poniżej został zmieniony numer albumu, ale pozostałe atrybuty były puste dlatego też nie ma ich w zaaktualizowanym rekordzie. Przykład użycia pretty().

```
Kod:
```

```
db.myData.update({no_album: '400000'},{no_album: '000000'})
```

Zrzut ekranu:

```
controlled.

o.myOata.update((no.album: '400000'), (no_albuteResult(( "nMatched" : 1, "nUpserted" : 0, "
).myOata.find((no.album: '400000')).pretty()
o.myOata.find((no_album: '000000')).pretty()
o.myOata.find((no_album: '000000')).pretty()
id" : ObjectId("60b1071d656d9e18a0bf81af"),
```

W rekordzie zostanie zmienione tylko jedno pole, a reszta zostaje taka sama. ii.

```
db.myData.update({no_album: '400012'},{$set: {no_album: '500000'}})
```

Zrzut ekranu:

```
db.myData.update((no_album: '400012'),{$set: {no_album: '500000'}})
iteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
db.myData.find((no_album: '50000')).pretty()
"_id" : ObjectId('60b1071d656d9e18a0bf81af"), "no_album" : "500000'
                          '_id" : ObjectId("60b1082b656d9e18a0bf81b1"),
'no album" : "500000",
                           _or_olyecting Todios20656
no_album': "S000000",
name": "Lukasz",
birth_date": "01.07.2000",
pesell: "12345421455",
address": "Krakow 5",
phone_number": "222222222",
mail_address": "aba@02.pl",
field_of_study": "IO",
faculty": "WIMIIR"
```

d. Delete - usunięcie rekordu

```
db.myData.remove({no album: '500000'})
```

Zrzut ekranu:

```
ta.find({no_album: '500000'}).pretty()
ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "500000" }
         _id" : ObjectId("60b1082b656d9e18a0bf81b1"),
             _album": "S00000",

_album": "S00000",

me": "Lukasz",
roname": "Gra",
rth_date": "01.07.2000",
sel": "12245421455",
dress": "Krakow 5",
one_number": "22222222",
il_address": "aba@o2.pl",
eld_of_study": "IO",
culty": "WIMIIR"
yData.remove({no_album: '500000'})
esult({ "nRemoved" : 2 })
yData.find({no_album: '500000'}).pretty()
```

4. Operator \$text

Operator ten służy do wyszukiwania tekstowego z indeksem tekstowym.

a. Aby on działał należał wpierw stworzyć ten indeks:

```
db.myData.createIndex( {no_album: 'text', name: 'text', surname: 'text',
birth_date: 'text', pesel: 'text', address:'text', phone_number: 'text',
mail_address: 'text', field_of_study: 'text', faculty: 'text'})
```

*zamiast wypisywania wszystkich atrybutów można było użyć \$**.

Zrzut ekranu:

```
a.createIndex( {no_album: 'text', name: 'text', surname: 'text', birth_date: 'text
ne_number: 'text', mail_address: 'text', field_of_study: 'text', faculty: 'text'})
    NIndexesBefore" : 2,
NIndexesAfter" : 2,
Ne" : "all indexes already exist",
```

b. Następnie można było go już używać, np. wyszukanie tekstu WIMiIP:

Kod:

```
db.myData.find({$text: {$search: 'WIMiIP'}})
```

Zrzut ekranu:

```
.IZUI EKKAITU.
| db.myData.find({$text: {$search: 'WIMiIP'}})
| _id" : ObjectId("60b107f4656d9e18a0bf81b0"), "no_album" : "400011", "name" : "Mateusz", "surname" : "Gryn", "birth_d
| e" : "20.09.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone_number" : "111111111", "mail_address" : "yy
| o2.pl", "field_of_study" : "IT", "faculty" : "WIMiIP" }
```

Zapytania MongoDB

1. Wczytanie do nowej bazy danych dane pochodzące z https://media.mongodb.org/zips.json

W tym celu użyto instrukcji mongoimport w cmd otworzonej jako administrator. Jednakże przed tym należało pobrać narzedzia MongoDB ze strony: https://www.mongodb.com/try/download/databasetools?tck=docs databasetools, a następnie przenieść je do folderu bin w MongoDB/Server/4.4. Plik zawierający dane również ma sie powinien sie tam znaleźć. Po tym należało wejść w cmd również do tego katalogu i wykonać poniższą instrukcje:

mongoimport /db:nowa baza /collection:zapytania --file=zips.json

Zrzut ekranu:

```
C:\Program Files\MongoDB\Ser
2021-05-28T19:37:46.983+0200
                                    \4.4\bin>mongoimport
                                     connected to: mongodb://localhost/
021-05-28T19:37:47.405+0200
                                     29353 document(s) imported successfully. 0 document(s) failed to import
```

Potwierdzenie dodania danych:

2. Zapytania z różnymi operatorami

l.p.	Co zapytanie powinno zrobić?	Kod	Wyniki – screeny pod tabelą
1	Wyświetlić sumę, średnią, minimalną, maksymalną, pierwszą i ostatnią wartość zmiennej pop	<pre>db.zapytania.aggregate([{\$group:{ _id: null, "sum": {\$sum: "\$pop"}, "avg": {\$avg: "\$pop"}, "min": {\$min: "\$pop"}, "max": {\$max: "\$pop"}, "first": {\$first: "\$pop"}, "last": {\$last: "\$pop"}}])</pre>	Spełnia początkowe założenia
2	Wyświetlenie rekordów gdzie id!=01081 i pop>100000 i pop<112000	<pre>db.zapytania.find({\$and: [{"_id":{\$ne:"01081"}}, {"pop":{\$gt: 100000}}, {"pop":{\$lt: 112000}}]))</pre>	Spełnia początkowe założenia
3	To samo co powyżej, ale z zamkniętymi przedziałami i operatorem logicznym or: suma przedziałów	<pre>db.zapytania.find({\$or: [{"_id":{\$ne:"01081"}}, {"pop":{\$gte: 100000}}, {"pop":{\$lte: 112000}}]))</pre>	Spełnia początkowe założenia
4	Wyświetlenie rekordów, które mają odwrotne warunki do zapytania 2	<pre>db.zapytania.find({\$nor: [{\$and: [{"_id":{\$ne:"01081"}}, {"pop":{\$gte: 100000}}, {"pop":{\$lte: 112000}}]}]})</pre>	Spełnia początkowe założenia
5	Wyświetlenie rekordów, które spełniają warunki tak jak w zapytaniu 2, ale zamiast pop>100000 jest pop<100000	<pre>db.zapytania.find({\$or: [{"_id":{\$ne:"01081"}}, {"pop":{\$not: {\$gte: 100000}}}, {"pop":{\$lte: 112000}}]})</pre>	Spełnia początkowe założenia
6	Wyświetlenie pierwszego rekordu zwróconego przez zapytanie 2	<pre>db.zapytania.find({\$and: [{"_id":{\$ne:"01081"}}, {"pop":{\$gt: 100000}}, {"pop":{\$lt: 112000}}]}).limit(1)</pre>	Spełnia początkowe założenia
7	Wyświetlenie rekordów zwróconych przez zapytanie 2 z pominięciem pierwszych 2	<pre>db.zapytania.find({\$and: [{"_id":{\$ne:"01081"}}, {"pop":{\$gt: 100000}}, {"pop":{\$lt: 112000}}]}).skip(2)</pre>	Spełnia początkowe założenia
8	Wyświetlenie rekordów ze stanem MA za pomocą regexa(wyrażenie regularne)	<pre>db.zapytania.find({ state: { \$regex: /MA\$/ } })</pre>	Spełnia początkowe założenia

1:
> db.zapytania.aggregate([{\$group:{_id:null, "sum": {\$sum: "\$pop"}, "avg": {\$avg: "\$pop"}, "min": {\$min: "\$pop"}, "max": {\$max: "\$pop"}, "first": {\$first: "\$pop"}, "last": {\$last: "\$pop"}}}]}]
(" 'id" : null "sum" - 248408400 "avg" - 8467 79426/2937348 "min" - 9 "max" - 112947 "first" - 177 "last" - 422 \\ "1284 " - 1784 " - 1884 "

```
2:

> db.zapytania.find({$and: [{"_id":{$ne:"01081"}}, {"pop":{$gt: 100000}}, {"pop":{$lt: 112000}}])

("_id": "10021", "city": "NEW YORK", "loc": [ -73.958805, 40.768476], "pop": 106564, "state": "NY" }

{"_id": "10025", "city": "leW YORK", "loc": [ -73.958312, 40.797466], "pop": 100027, "state": "NY" }

{"_id": "11226", "city": "BROOKLYN", "loc": [ -73.956985, 40.646694], "pop": 111396, "state": "NY" }
```

```
3.

> db.zapytania.find(($or: [("_id":($ne:"01081")), ("pop":($te: 100000)), ("pop":($tle: 112000))]))

("_id": "01012", "city": "CHESTERFIELD", "loc": [ -72.833309, 42.38167 ], "pop": 177, "state": "MA" )

("_id": "01008", "city": "BLANDFORD", "loc": [ -72.9835109, 42.38167 ], "pop": 1240, "state": "MA" )

("_id": "010011", "city": "CHESTERFIELD", "loc": [ -72.988761, 42.182209 ], "pop": 1268, "state": "MA" )

("_id": "01002", "city": "CHESTERFIELD", "loc": [ -72.988761, 42.17941 ], "pop": 1688, "state": "MA" )

("_id": "01002", "city": "CHICOPRE", "loc": [ -72.98165, 42.377017 ], "pop": 31095, "state": "MA" )

("_id": "01013", "city": "CHICOPRE", "loc": [ -72.575162, 42.176403 ], "pop": 1308, "state": "MA" )

("_id": "01012", "city": "MUNIT TOW, "loc": [ -72.667992, 42.163206 ], "pop": 1308, "state": "MA" )

("_id": "01022", "city": "MUSTITOR AFB", "loc": [ -72.575174, 42.196672 ], "pop": 1704, "state": "MA" )

("_id": "01028", "city": "EASTIONERADOM", "loc": [ -72.95767, 42.435296 ], "pop": 1484, "state": "MA" )

("_id": "01038", "city": "EASTIONERADOM", "loc": [ -72.95767, 42.435296 ], "pop": 1985, "state": "MA" )

("_id": "01031", "city": "GILBERTUTILL", "loc": [ -72.675077, 42.07182 ], "pop": 11985, "state": "MA" )

("_id": "01031", "city": "GILBERTUTILL", "loc": [ -72.968793, 42.332194 ], "pop": 1285, "state": "MA" )

("_id": "01034", "city": "TOLLAND", "loc": [ -72.968793, 42.96234 ], "pop": 1652, "state": "MA" )

("_id": "01038", "city": "TOLLAND", "loc": [ -72.968793, 42.96234 ], "pop": 1652, "state": "MA" )

("_id": "01038", "city": "HATFELD", "loc": [ -72.66175, 42.076234 ], "pop": 1652, "state": "MA" )

("_id": "01038", "city": "HATFELD", "loc": [ -72.66175, 42.06275, 42.06923 ], "pop": 1385, "state": "MA" )

("_id": "01038", "city": "HATFELD", "loc": [ -72.66175, 42.06207, 42.06207 ], "pop": 1387, "state": "MA" )

("_id": "01038", "city": "HATFELD", "loc": [ -72.66175, 42.06207, 42.0607, 50, pop": 1387, "state": "MA" )

("_id": "01038", "city": "HATFELD", "loc": [ -72.661673, 42.02607 ], "pop": 13
```

```
۵٠
                                                                                                                                                                                                                                                                                                                           c: [{$and: [{"_id":{$ne:"01081"}}, {"pop":{$gte: 100000}}, {"pop":{$lte: 112000}}]]])}
": "CHESTERFIELD", "loc": [ -72.833309, 42.38167 ], "pop": 177, "state": "MA" }
": "BLANDFORD", "loc": [ -72.936114, 42.182949 ], "pop": 1240, "state": "MA" }
": "CHESTER", "loc": [ -72.936114, 42.182949 ], "pop": 1688, "state": "MA" }
": "CUSHMAN", "loc": [ -72.5786142, 42.179421 ], "pop": 1688, "state": "MA" }
": "CUSHMAN", "loc": [ -72.576142, 42.176443 ], "pop": 36963, "state": "MA" }
": "CHICOPEE", "loc": [ -72.5796142, 42.176443 ], "pop": 23396, "state": "MA" }
": "MOUNT TOM", "loc": [ -72.679921, 42.264319 ], "pop": 12386, "state": "MA" }
": "MOUNT TOM", "loc": [ -72.679921, 42.264319 ], "pop": 16864, "state": "MA" }
": "WESTOVER AFB", "loc": [ -72.595655, 42.367203 ], "pop": 1764, "state": "MA" }
": "GILBERTVILLE", "loc": [ -72.5758657, 42.345296 ], "pop": 11985, "state": "MA" }
": "GILBERTVILLE", "loc": [ -72.675077, 42.07182 ], "pop": 11985, "state": "MA" }
": "GOSHEN", "loc": [ -72.844092, 42.466234 ], "pop": 122, "state": "MA" }
": "GOSHEN", "loc": [ -72.520001, 42.255704 ], "pop": 15526, "state": "MA" }
": "GRANBV", "loc": [ -72.520001, 42.255704 ], "pop": 15526, "state": "MA" }
": "HAMFIELD", "loc": [ -72.520001, 42.255704 ], "pop": 1570, "state": "MA" }
": "HAMFIELD", "loc": [ -72.520001, 42.255704 ], "pop": 4709, "state": "MA" }
": "HAMFIELD", "loc": [ -72.520001, 42.255704 ], "pop": 4709, "state": "MA" }
": "HAMFIELD", "loc": [ -72.520001, 42.255704 ], "pop": 4709, "state": "MA" }
": "HAMFIELD", "loc": [ -72.520001, 42.255704 ], "pop": 4709, "state": "MA" }
": "HAMPOENVILLE", "loc": [ -72.6263193, 42.202007 ], "pop": 4709, "state": "MA" }
": "HAMFIELD", "loc": [ -72.5262739, 42.30406 ], "pop": 1338, "state": "MA" }
": "HAGOLWARM, 
                                                                                                                         Cania. Tino (Shor):
"01012", "city":
"01008", "city":
"01008", "city":
"01002", "city":
"01002", "city":
"01013", "city":
"01022", "city":
"01022", "city":
"01028", "city":
"01028", "city":
"01028", "city":
"01031", "city":
"01031", "city":
"01031", "city":
"01033", "city":
"01033", "city":
"01033", "city":
"01034", "city":
"01034", "city":
"01036", "city":
"01036", "city":
"01037", "city":
"01037", "city":
"01039", "city":
                                                                                                                                "01012",
"01008",
"01011",
                       "_id" :
                                                 id"
                                          _id"
_id"
_id"
_id"
                                              id"
   5:
                                                                                                                                                                                                                                                                                                                                                      [{"_id":{$ne:"01081"}}, {"pop":{$not: {$gte: 100000}}}, {"pop":{$lte: 112000}}]})

"GHESTERFIELD", "loc": [ -72.833309, 42.83167 ], "pop": 177, "state": "MA" }

"BLANDFORD", "loc": [ -72.988761, 42.279421 ], "pop": 177, "state": "MA" }

"CHESTER", "loc": [ -72.988761, 42.279421 ], "pop": 1688, "state": "MA" }

"CHESTER", "loc": [ -72.51565, 42.377017 ], "pop": 36963, "state": "MA" }

"CHICOPEE", "loc": [ -72.576142, 42.176443 ], "pop": 31495, "state": "MA" }

"GHICOPEE", "loc": [ -72.576142, 42.162046 ], "pop": 31495, "state": "MA" }

"MOUNT TOM", "loc": [ -72.679921, 42.264319 ], "pop": 16864, "state": "MA" }

"MESTOVER AFB", "loc": [ -72.595567, 42.495296 ], "pop": 1764, "state": "MA" }

"GENTIONGHEADOW", "loc": [ -72.905767, 42.495296 ], "pop": 1384, "state": "MA" }

"FEEDING HILLS", "loc": [ -72.679921, 42.967203 ], "pop": 13987, "state": "MA" }

"GILBERTVILLE", "loc": [ -72.198585, 42.332194 ], "pop": 1385, "state": "MA" }

"GOSSHEN", "loc": [ -72.908793, 42.060720 ], "pop": 1652, "state": "MA" }

"GANNBY", "loc": [ -72.590061, 42.255704 ], "pop": 1652, "state": "MA" }

"HAFTFIELD:, "loc": [ -72.66193, 42.36276 ], "pop": 3184, "state": "MA" }

"HAPTPEND:, "loc": [ -72.66193, 42.20007 ], "pop": 3184, "state": "MA" }

"HAPYDOKE", "loc": [ -72.66193, 42.202007 ], "pop": 43704, "state": "MA" }

"HAPYDOKI", "loc": [ -72.622739, 42.070206 ], "pop": 15338, "state": "MA" }

"AGAMAM", "loc": [ -72.622739, 42.070206 ], "pop": 15338, "state": "MA" }

"AGAMAM", "loc": [ -72.622739, 42.070206 ], "pop": 15338, "state": "MA" }
                                                                                                                            "01012",
"01008",
"01011",
"01011",
"0102",
"01022",
"01022",
"01028",
"01032",
"01032",
"01032",
"01033",
"01038",
"01038",
"01038",
"01039",
"01039",
"01039",
"01040",
"01039",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"01040",
"010
   6:
                                                                                     zapytania.find({$and: [{"_id":{$ne:"01081"}}, {"pop":{$gt: 100000}}, {"pop":{$lt: 112000}}]})
d" : "10021", "city" : "NEW YORK", "loc" : [ -73.958805, 40.768476 ], "pop" : 106564, "state"
                                                       id" : "10021",
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  "state
                             db.zapytania.find({$and: [{"_id":{$ne:"01081"}}, {"pop":{$gt: 100000}}, {"pop":{$lt: 112000}}]}).skip(2
"_id": "11226", "city": "BROOKLYN", "loc": [ -73.956985, 40.646694 ], "pop": 111396, "state": "NY"
8:
                                                                                                                                                                                                                                                                                                                                                                                                                    "CHESTERFIELD", "loc": [ -72.833309, 42.38167 ], "pop": 177, "state": "MA" }
"BLANDFORD", "loc": [ -72.936114, 42.182949 ], "pop": 1240, "state": "MA" }
"CHESTER", "loc": [ -72.938761, 42.279421 ], "pop": 1688, "state": "MA" }
"CUSHMAN", "loc": [ -72.51565, 42.377017 ], "pop": 36963, "state": "MA" }
"CHICOPEE", "loc": [ -72.576142, 42.176443 ], "pop": 31495, "state": "MA" }
"CHICOPEE", "loc": [ -72.607962, 42.162046 ], "pop": 23396, "state": "MA" }
"MOUNIT TOM", "loc": [ -72.679921, 42.264319 ], "pop": 16864, "state": "MA" }
"WESTOVER AFB", "loc": [ -72.578657, 42.19672 ], "pop": 1764, "state": "MA" }
"CUMMINGTON", "loc": [ -72.905767, 42.435296 ], "pop": 1764, "state": "MA" }
"EAST LONGMEADOW", "loc": [ -72.505565, 42.067203 ], "pop": 13367, "state": "MA" }
"FEEDING HILLS", "loc": [ -72.675977, 42.07182 ], "pop": 11985, "state": "MA" }
"GILBERTVILLE", "loc": [ -72.198585, 42.332194 ], "pop": 12355, "state": "MA" }
"GOSHEN", "loc": [ -72.984793, 42.070234 ], "pop": 122, "state": "MA" }
"GRANBY", "loc": [ -72.968793, 42.070234 ], "pop": 15526, "state": "MA" }
"HAMTFIELD", "loc": [ -72.431823, 42.364756 ], "pop": 4709, "state": "MA" }
"HAYPDEN", "loc": [ -72.616735, 42.381799 ], "pop": 1387, "state": "MA" }
"HAYPOEN", "loc": [ -72.793178, 42.381799 ], "pop": 15538, "state": "MA" }
"HAYPOENVILLE", "loc": [ -72.703178, 42.381799 ], "pop": 15338, "state": "MA" }
"AGAWAM", "loc": [ -72.622739, 42.070206 ], "pop": 15338, "state": "MA" }
                                                                                                                                             tania.find( { sta
"01012", "city"
"01008", "city"
"01011", "city"
"01002", "city"
"01020", "city"
"01027", "city"
"01022", "city"
"01028", "city"
"01038", "city"
"01032", "city"
"01032", "city"
"01032", "city"
"01033", "city"
"01033", "city"
"01033", "city"
"01033", "city"
"01033", "city"
"01038", "city"
"01038", "city"
"01039", "city"
"01039", "city"
"01039", "city"
"01039", "city"
                                                       id"
                                "_id"
                             "_id"
"_id"
                                "_id"
                                                           id"
                                                       _id"
                                                           id"
                                                       _id"
                                                           id"
                                                              id"
                                                              id"
                                                              id"
                                                                                                                                                                                                                                                                                       "city" :
```

WNIOSKI

Po wykonaniu tego ćwiczenia można stwierdzić, że baza danych MongoDB jest przyjemna w obsłudzę, łatwa w użyciu i mało zagmatwana. Nie trzeba wstawiać typów zmiennych w kolekcji i pilnować ilości atrybutów co bardzo wspomaga pracę z bazą danych. Funkcję, które ona udostępnia również są łatwe do stworzenia i bardzo logiczne przez co nie sprawiają dużych problemów.

MongoDB w prównaniu z MySQL i Oraclem wydaje się być łatwiejsza w obsłudzę. Można odnieść wrażenie, że robi ona automatycznie wiele instrukcji zamiast obarczać tym użytkownika. Wrażenie to może po prostu wynikać, z tego że jest to nierelacyjna baza danych przez co dużo instrukcji, które należało wykonać np. w MySQL są zbędne przy czym wcześniej pracowano tylko z relacyjnymi bazami danych, gdzie ich wystąpienia były konieczne.

Niestety MongoDB, co było wspomniane wcześniej, jest nierelacyjną bazą danych dlatego nie ma scheme. Wszystkie dane są zebrane w jednym miejscu co w pracy nad dużymi bazami może być wielkim utrudnieniem. W takiej sytuacji dużo lepszą opcją będzie wybór baz relacyjnych.