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

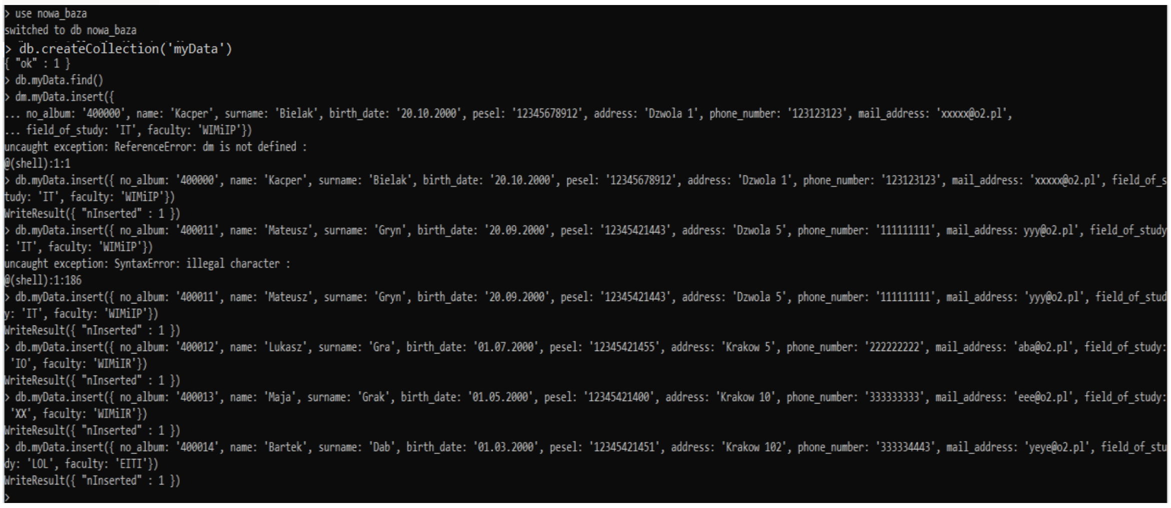
Podstawy MongoDB

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

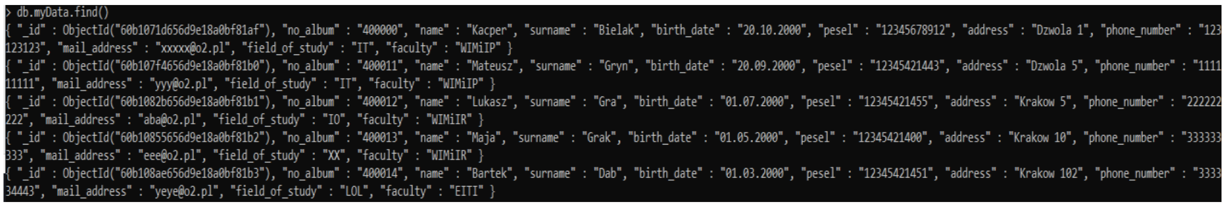
Kod:

```
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:
'111111111', 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:
'222222222', 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:
'333333333', 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:



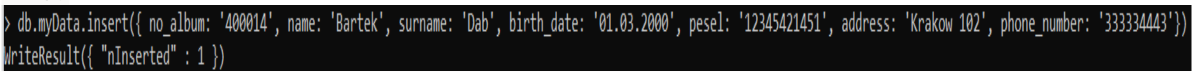
```
> use nowa_baza
switched to db nowa_baza
> db.createCollection('myData')
{ "ok" : 1 }
> db.myData.find()
> dm.myData.insert({
... no_album: '400000', name: 'Kacper', surname: 'Bielak', birth_date: '20.10.2000', pesel: '12345678912', address: 'Dzwola 1', phone_number: '123123123', mail_address: 'xxxxx@o2.pl',
... field_of_study: 'IT', faculty: 'WIMiIP'})
uncaught exception: ReferenceError: dm is not defined :
@(shell):1:1
> 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_s
tudy: 'IT', faculty: 'WIMiIP'})
WriteResult({ "nInserted" : 1 })
> db.myData.insert({ no_album: '400011', name: 'Mateusz', surname: 'Gryn', birth_date: '20.09.2000', pesel: '12345421443', address: 'Dzwola 5', phone_number: '111111111', mail_address: yyy@o2.pl', field_of_study
: 'IT', faculty: 'WIMiIP'})
uncaught exception: SyntaxError: illegal character :
@(shell):1:186
> db.myData.insert({ no_album: '400011', name: 'Mateusz', surname: 'Gryn', birth_date: '20.09.2000', pesel: '12345421443', address: 'Dzwola 5', phone_number: '111111111', mail_address: 'yyy@o2.pl', field_of_stud
y: 'IT', faculty: 'WIMiIP'})
WriteResult({ "nInserted" : 1 })
> db.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:
 'IO', faculty: 'WIMiIR'})
WriteResult({ "nInserted" : 1 })
> db.myData.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:
 'XX', faculty: 'WIMiIR'})
WriteResult({ "nInserted" : 1 })
> 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_stu
dy: 'LOL', faculty: 'EITI'})
WriteResult({ "nInserted" : 1 })
>
```



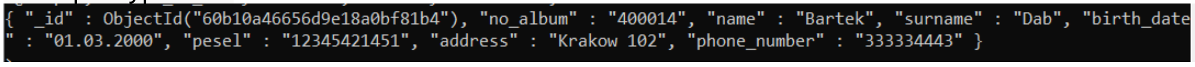
```
> db.myData.find()
{ "_id" : ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "400000", "name" : "Kacper", "surname" : "Bielak", "birth_date" : "20.10.2000", "pesel" : "12345678912", "address" : "Dzwola 1", "phone_number" : "123
123123", "mail_address" : "xxxxx@o2.pl", "field_of_study" : "IT", "faculty" : "WIMiIP" }
{ "_id" : ObjectId("60b107f4656d9e18a0bf81b0"), "no_album" : "400011", "name" : "Mateusz", "surname" : "Gryn", "birth_date" : "20.09.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone_number" : "1111
11111", "mail_address" : "yyy@o2.pl", "field_of_study" : "IT", "faculty" : "WIMiIP" }
{ "_id" : ObjectId("60b1082b656d9e18a0bf81b1"), "no_album" : "400012", "name" : "Lukasz", "surname" : "Gra", "birth_date" : "01.07.2000", "pesel" : "12345421455", "address" : "Krakow 5", "phone_number" : "222222
222", "mail_address" : "aba@o2.pl", "field_of_study" : "IO", "faculty" : "WIMiIR" }
{ "_id" : ObjectId("60b10855656d9e18a0bf81b2"), "no_album" : "400013", "name" : "Maja", "surname" : "Grak", "birth_date" : "01.05.2000", "pesel" : "12345421400", "address" : "Krakow 10", "phone_number" : "333333
333", "mail_address" : "eee@o2.pl", "field_of_study" : "XX", "faculty" : "WIMiIR" }
{ "_id" : ObjectId("60b108ae656d9e18a0bf81b3"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "3333
34443", "mail_address" : "yeye@o2.pl", "field_of_study" : "LOL", "faculty" : "EITI" }
```

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:


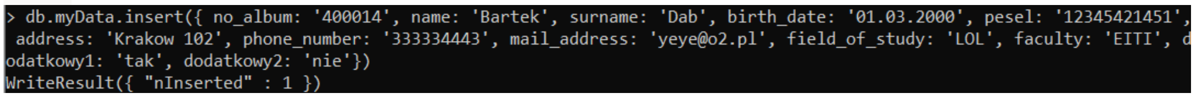
```
> db.myData.insert({ no_album: '400014', name: 'Bartek', surname: 'Dab', birth_date: '01.03.2000', pesel: '12345421451', address: 'Krakow 102', phone_number: '333334443'})
WriteResult({ "nInserted" : 1 })
```

   Efekt po wypisaniu:


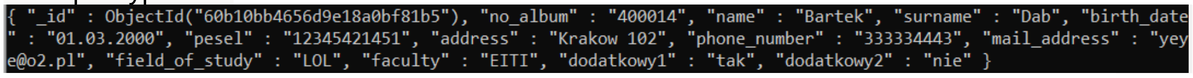
```
{ "_id" : ObjectId("60b10a46656d9e18a0bf81b4"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443" }
```

   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', d
odatkowy1: 'tak', dodatkowy2: 'nie'})
WriteResult({ "nInserted" : 1 })
```

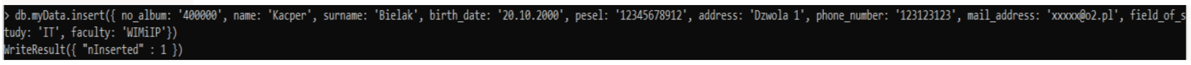
   Efekt po wypisaniu:


```
{ "_id" : ObjectId("60b10bb4656d9e18a0bf81b5"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443", "mail_address" : "yey
e@o2.pl", "field_of_study" : "LOL", "faculty" : "EITI", "dodatkowy1" : "tak", "dodatkowy2" : "nie" }
```

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

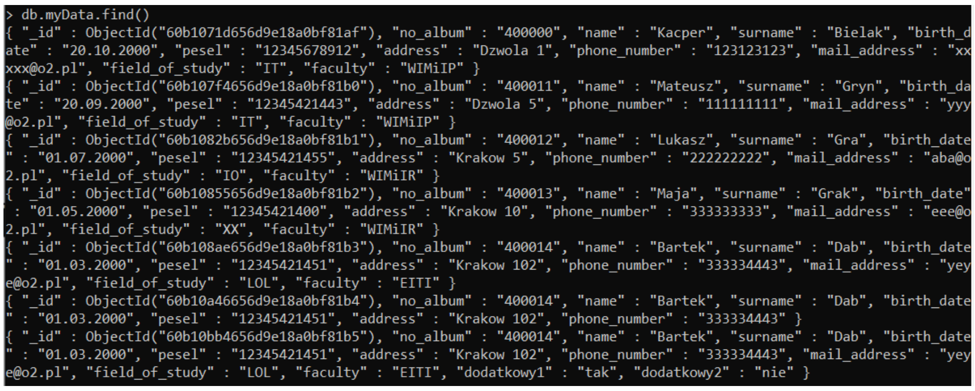
3. **CRUD**

   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.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_s
tudy: 'IT', faculty: 'WIMiIP'})
WriteResult({ "nInserted" : 1 })
```

b. **Read – wyświetlenie wszystkich rekordów**
Kod:
`db.myData.find()`

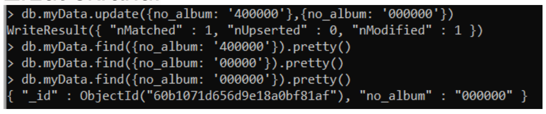
Zrzut ekranu:


```
> db.myData.find()
{ "_id" : ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "400000", "name" : "Kacper", "surname" : "Bielak", "birth_d
ate" : "20.10.2000", "pesel" : "12345678912", "address" : "Dzwola 1", "phone_number" : "123123123", "mail_address" : "xx
xxx@o2.pl", "field_of_study" : "IT", "faculty" : "WIMiIP" }
{ "_id" : ObjectId("60b107f4656d9e18a0bf81b0"), "no_album" : "400011", "name" : "Mateusz", "surname" : "Gryn", "birth_da
te" : "20.09.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone_number" : "111111111", "mail_address" : "yyy
@o2.pl", "field_of_study" : "IT", "faculty" : "WIMiIP" }
{ "_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" : "aba@o
2.pl", "field_of_study" : "IO", "faculty" : "WIMiIR" }
{ "_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@o
2.pl", "field_of_study" : "XX", "faculty" : "WIMiIR" }
{ "_id" : ObjectId("60b108ae656d9e18a0bf81b3"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443", "mail_address" : "yey
e@o2.pl", "field_of_study" : "LOL", "faculty" : "EITI" }
{ "_id" : ObjectId("60b10a46656d9e18a0bf81b4"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443" }
{ "_id" : ObjectId("60b10bb4656d9e18a0bf81b5"), "no_album" : "400014", "name" : "Bartek", "surname" : "Dab", "birth_date
" : "01.03.2000", "pesel" : "12345421451", "address" : "Krakow 102", "phone_number" : "333334443", "mail_address" : "yey
e@o2.pl", "field_of_study" : "LOL", "faculty" : "EITI", "dodatkowy1" : "tak", "dodatkowy2" : "nie" }
```

c. **Update – zmiana numer albumu studenta**
Update można wykonać na dwa sposoby:

i. 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().**

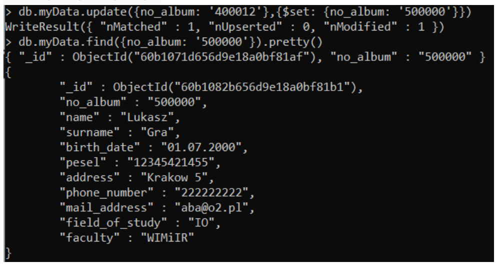
Kod:
`db.myData.update({no_album: '400000'},{no_album: '000000'})`

Zrzut ekranu:


```
> db.myData.update({no_album: '400000'},{no_album: '000000'})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.myData.find({no_album: '400000'}).pretty()
> db.myData.find({no_album: '00000'}).pretty()
> db.myData.find({no_album: '000000'}).pretty()
{ "_id" : ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "000000" }
```

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

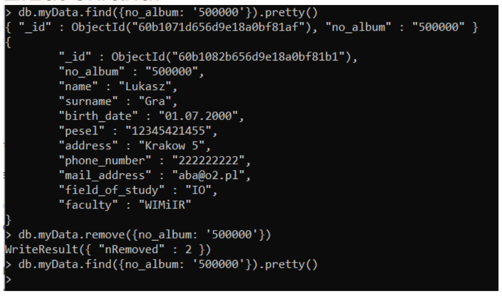
Kod:
`db.myData.update({no_album: '400012'},{$set: {no_album: '500000'}})`

Zrzut ekranu:


```
> db.myData.update({no_album: '400012'},{$set: {no_album: '500000'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.myData.find({no_album: '500000'}).pretty()
{ "_id" : ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "500000" }
{
        "_id" : ObjectId("60b1082b656d9e18a0bf81b1"),
        "no_album" : "500000",
        "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" : "IO",
        "faculty" : "WIMiIR"
}
```

d. **Delete – usunięcie rekordu**

Kod:
`db.myData.remove({no_album: '500000'})`

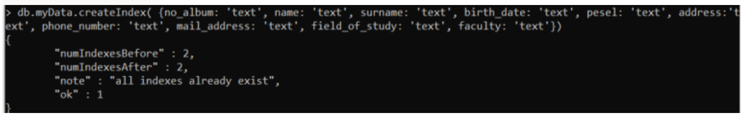
Zrzut ekranu:


```
> db.myData.find({no_album: '500000'}).pretty()
{ "_id" : ObjectId("60b1071d656d9e18a0bf81af"), "no_album" : "500000" }
{
        "_id" : ObjectId("60b1082b656d9e18a0bf81b1"),
        "no_album" : "500000",
        "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" : "IO",
        "faculty" : "WIMiIR"
}
> db.myData.remove({no_album: '500000'})
WriteResult({ "nRemoved" : 2 })
> db.myData.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:
      Kod:
      `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'})`

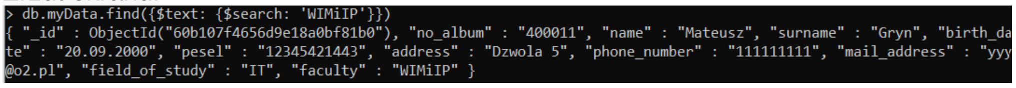
      *zamiast wypisywania wszystkich atrybutów można było użyć $**.

      Zrzut ekranu:


```
> db.myData.createIndex( {no_album: 'text', name: 'text', surname: 'text', birth_date: 'text', pesel: 'text', address:'t
ext', phone_number: 'text', mail_address: 'text', field_of_study: 'text', faculty: 'text'})
{
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 2,
        "note" : "all indexes already exist",
        "ok" : 1
}
```

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

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

      Zrzut ekranu:


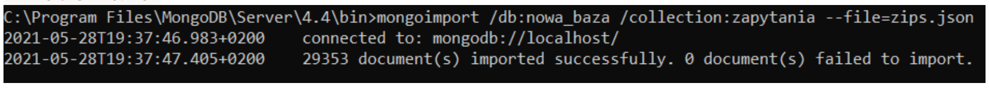
```
> db.myData.find({$text: {$search: 'WIMiIP'}})
{ "_id" : ObjectId("60b107f4656d9e18a0bf81b0"), "no_album" : "400011", "name" : "Mateusz", "surname" : "Gryn", "birth_da
te" : "20.09.2000", "pesel" : "12345421443", "address" : "Dzwola 5", "phone_number" : "111111111", "mail_address" : "yyy
@o2.pl", "field_of_study" : "IT", "faculty" : "WIMiIP" }
```

Zapytania MongoDB

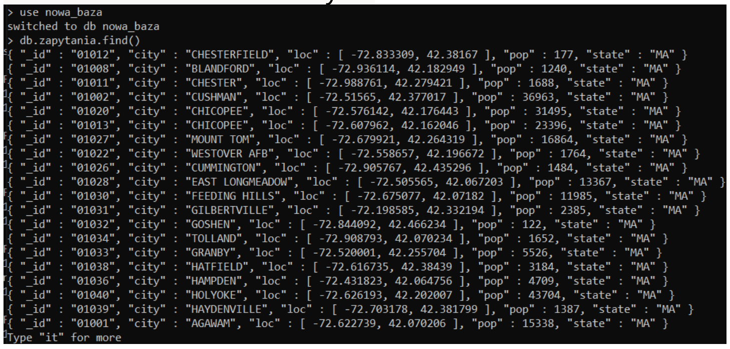
1. **Wczytanie do nowej bazy danych dane pochodzące z https://media.mongodb.org/zips.json**

   W tym celu użyto instrukcji mongoimport w cmd otworzonej jako administrator. Jednakże przed tym należało pobrać narzędzia MongoDB ze strony: https://www.mongodb.com/try/download/database-tools?tck=docs_databasetools, a następnie przenieść je do folderu bin w MongoDB/Server/4.4. Plik zawierający dane również ma się powinien się 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\Server\4.4\bin>mongoimport /db:nowa_baza /collection:zapytania --file=zips.json
2021-05-28T19:37:46.983+0200    connected to: mongodb://localhost/
2021-05-28T19:37:47.405+0200    29353 document(s) imported successfully. 0 document(s) failed to import.
```

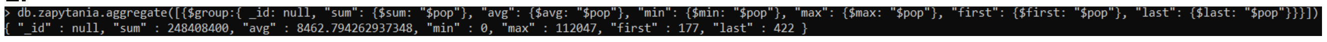
   Potwierdzenie dodania danych:


```
> use nowa_baza
switched to db nowa_baza
> db.zapytania.find()
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }
{ "_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }
{ "_id" : "01026", "city" : "CUMMINGTON", "loc" : [ -72.905767, 42.435296 ], "pop" : 1484, "state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA" }
{ "_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state" : "MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }
{ "_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state" : "MA" }
{ "_id" : "01040", "city" : "HOLYOKE", "loc" : [ -72.626193, 42.202007 ], "pop" : 43704, "state" : "MA" }
{ "_id" : "01039", "city" : "HAYDENVILLE", "loc" : [ -72.703178, 42.381799 ], "pop" : 1387, "state" : "MA" }
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
Type "it" for more
```

2. Zapytania z różnymi operatorami

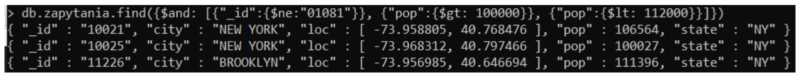
l.p.	Co zapytanie powinno zrobić?	Kod	Wyniki – screeny pod tabelą
1	Wyświetlić sumę, średnią, minimalną, maksymalną, pierwszą i ostatnią wartość zmiennej pop	`db.zapytania.aggregate([{$group:{ _id: null, "sum": {$sum: "$pop"}, "avg": {$avg: "$pop"}, "min": {$min: "$pop"}, "max": {$max: "$pop"}, "first":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

1:



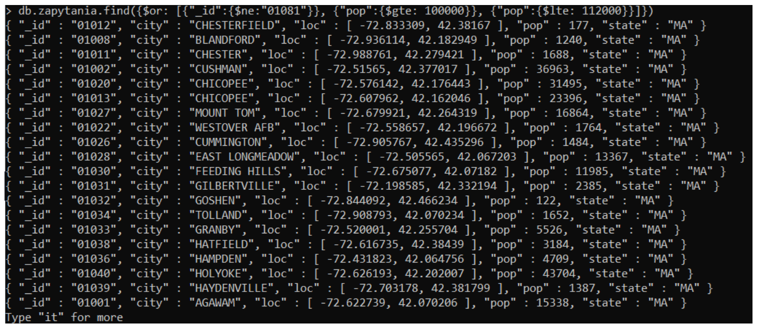
```
> db.zapytania.aggregate([{$group:{ _id: null, "sum": {$sum: "$pop"}, "avg": {$avg: "$pop"}, "min": {$min: "$pop"}, "max": {$max: "$pop"}, "first": {$first: "$pop"}, "last": {$last: "$pop"}}}])
{ "_id" : null, "sum" : 248408400, "avg" : 8462.794262937348, "min" : 0, "max" : 112047, "first" : 177, "last" : 422 }
```

2:



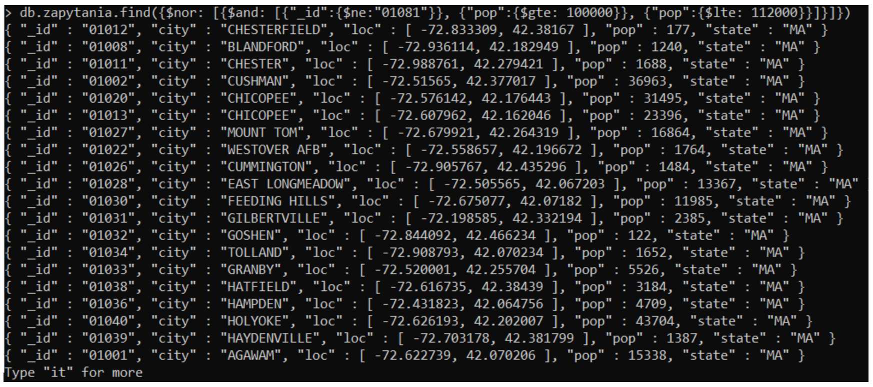
```
> db.zapytania.find({$and: [{"_id":{$ne:"01081"}}, {"pop":{$gt: 100000}}, {"pop":{$lt: 112000}}]})
{ "_id" : "10021", "city" : "NEW YORK", "loc" : [ -73.958805, 40.768476 ], "pop" : 106564, "state" : "NY" }
{ "_id" : "10025", "city" : "NEW YORK", "loc" : [ -73.968312, 40.797466 ], "pop" : 100027, "state" : "NY" }
{ "_id" : "11226", "city" : "BROOKLYN", "loc" : [ -73.956985, 40.646694 ], "pop" : 111396, "state" : "NY" }
```

3:



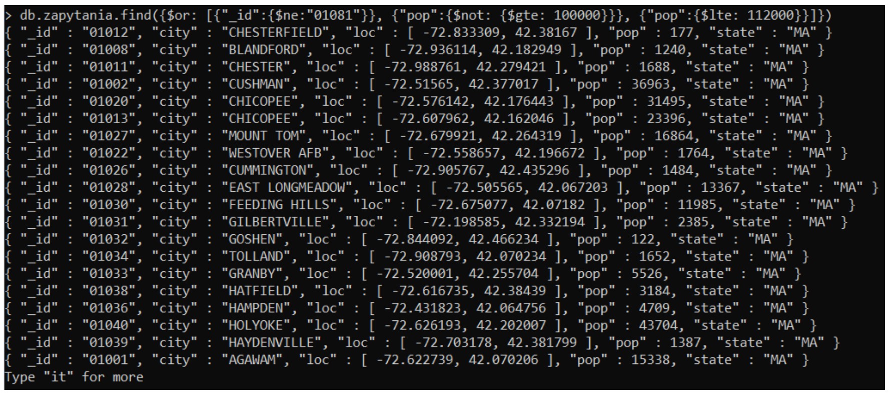
```
> db.zapytania.find({$or: [{"_id":{$ne:"01081"}}, {"pop":{$gte: 100000}}, {"pop":{$lte: 112000}}]})
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }
{ "_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }
{ "_id" : "01026", "city" : "CUMMINGTON", "loc" : [ -72.905767, 42.435296 ], "pop" : 1484, "state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA" }
{ "_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state" : "MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }
{ "_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state" : "MA" }
{ "_id" : "01040", "city" : "HOLYOKE", "loc" : [ -72.626193, 42.202007 ], "pop" : 43704, "state" : "MA" }
{ "_id" : "01039", "city" : "HAYDENVILLE", "loc" : [ -72.703178, 42.381799 ], "pop" : 1387, "state" : "MA" }
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
Type "it" for more
```

4:



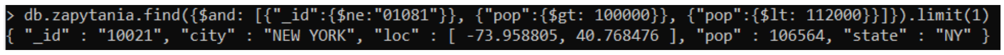
```
> db.zapytania.find({$nor: [{$and: [{"_id":{$ne:"01081"}}, {"pop":{$gte: 100000}}, {"pop":{$lte: 112000}}]}]})
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }
{ "_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }
{ "_id" : "01026", "city" : "CUMMINGTON", "loc" : [ -72.905767, 42.435296 ], "pop" : 1484, "state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA"
{ "_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state" : "MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }
{ "_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state" : "MA" }
{ "_id" : "01040", "city" : "HOLYOKE", "loc" : [ -72.626193, 42.202007 ], "pop" : 43704, "state" : "MA" }
{ "_id" : "01039", "city" : "HAYDENVILLE", "loc" : [ -72.703178, 42.381799 ], "pop" : 1387, "state" : "MA" }
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
Type "it" for more
```

5:



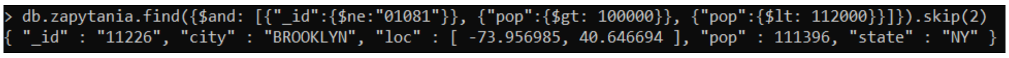
```
> db.zapytania.find({$or: [{"_id":{$ne:"01081"}}, {"pop":{$not: {$gte: 100000}}}, {"pop":{$lte: 112000}}]})
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }
{ "_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }
{ "_id" : "01026", "city" : "CUMMINGTON", "loc" : [ -72.905767, 42.435296 ], "pop" : 1484, "state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA" }
{ "_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state" : "MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }
{ "_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state" : "MA" }
{ "_id" : "01040", "city" : "HOLYOKE", "loc" : [ -72.626193, 42.202007 ], "pop" : 43704, "state" : "MA" }
{ "_id" : "01039", "city" : "HAYDENVILLE", "loc" : [ -72.703178, 42.381799 ], "pop" : 1387, "state" : "MA" }
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
Type "it" for more
```

6:



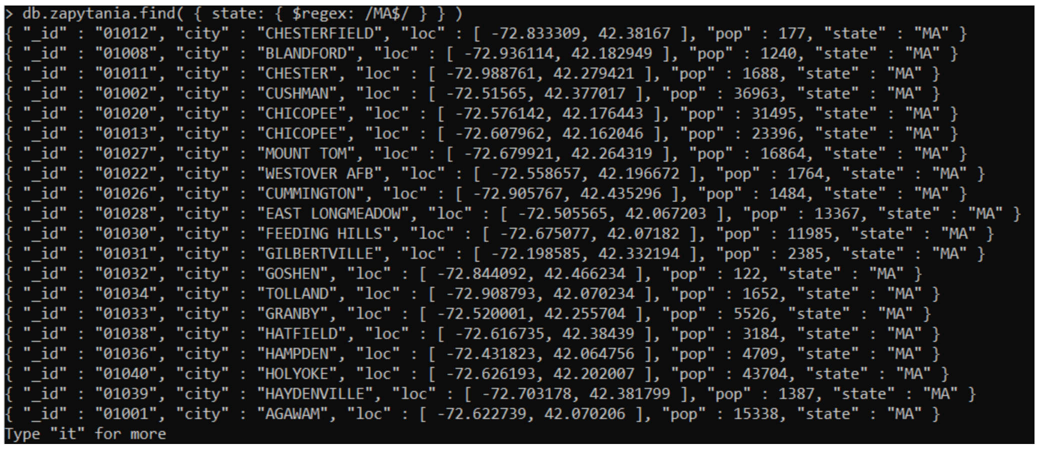
```
> db.zapytania.find({$and: [{"_id":{$ne:"01081"}}, {"pop":{$gt: 100000}}, {"pop":{$lt: 112000}}]}).limit(1)
{ "_id" : "10021", "city" : "NEW YORK", "loc" : [ -73.958805, 40.768476 ], "pop" : 106564, "state" : "NY" }
```

7:



```
> 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:



```
> db.zapytania.find( { state: { $regex: /MA$/ } } )
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }
{ "_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }
{ "_id" : "01026", "city" : "CUMMINGTON", "loc" : [ -72.905767, 42.435296 ], "pop" : 1484, "state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA" }
{ "_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state" : "MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }
{ "_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state" : "MA" }
{ "_id" : "01040", "city" : "HOLYOKE", "loc" : [ -72.626193, 42.202007 ], "pop" : 43704, "state" : "MA" }
{ "_id" : "01039", "city" : "HAYDENVILLE", "loc" : [ -72.703178, 42.381799 ], "pop" : 1387, "state" : "MA" }
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }
Type "it" for more
```

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.