

Mateusz Skowron

Zadanie 7

Stworzenie bazy:

use MS

Stworzenie kolekcji:

db.createCollection("student")

Struktura dokumentu:

```
{
  "indexNumber": (index number),
  "semester":    (semester number),
  "faculty":     (faculty shortcut like 'IET'),
  "firstName":   (first name),
  "lastName":    (last name),
  "age":         (age),
  "address": {
    "country":   (country name),
    "city":      (city name),
    "zipCode":   (zip-code like '38-203'),
    "street":    (street name),
    "homeNumber": (home number)
  },
  "subjects": [(subject name)],
  "grades": [{"subjectName": (subject name), "subjectGrades": [(grade)]}]
}
```

przykładowy dokument:

```
{
  "indexNumber": 405743,
  "semester": 4,
  "faculty": "IET",
  "firstName": "Mateusz",
  "lastName": "Skowron",
  "age": 20,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-203",
    "street": "Ulicowska",
    "homeNumber": 165
  },
  "subjects": ["math", "algorithms", "databases"],
  "grades": [
    {"subjectName": "math", "subjectGrades": [5.0, 4.5, 4.0]},
    {"subjectName": "algorithms", "subjectGrades": [3.0, 3.5, 3.0]},
    {"subjectName": "databases", "subjectGrades": [5.0, 5.0, 5.0]}
  ]
}
```

Wstawianie:

```
db.student.insertOne({
  "indexNumber": 405743,
  "semester": 4,
  "faculty": "IET",
  "firstName": "Mateusz",
  "lastName": "Skowron",
  "age": 20,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-203",
    "street": "Ulicowska",
    "homeNumber": 165
  },
  "subjects": ["math", "algorithms", "databases"],
  "grades": [
    { "subjectName": "math", "subjectGrades": [5.0, 4.5, 4.0] },
    { "subjectName": "algorithms", "subjectGrades": [3.0, 3.5, 3.0] },
    { "subjectName": "databases", "subjectGrades": [5.0, 5.0, 5.0] }
  ]
})
```

```
db.student.insertOne({
  "indexNumber": 405743,
  "semester": 4,
  "faculty": "IET",
  "firstName": "Mateusz",
  "lastName": "Skowron",
  "age": 20,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-203",
    "street": "Ulicowska",
    "homeNumber": 165
  },
  "subjects": ["math", "algorithms", "databases"],
  "grades": [
    { "subjectName": "math", "subjectGrades": [5.0, 4.5, 4.0] },
    { "subjectName": "algorithms", "subjectGrades": [3.0, 3.5, 3.0] },
    { "subjectName": "databases", "subjectGrades": [5.0, 5.0, 5.0] }
  ]
})
```

Shell Output

Shell Output (Documents) ×

← → → | 50 | Documents 1 to 1 | 🔍

```
{
  "acknowledged" : true,
  "insertedId" : ObjectId("6257078767213411a6c29d06")
}
```

```

db.student.insertOne({
  "indexNumber": 123123,
  "semester": 6,
  "faculty": "EAIIB",
  "firstName": "Rafal",
  "lastName": "Nieznany",
  "age": 22,
  "address": {
    "country": "Poland",
    "city": "Gdansk",
    "zipCode": "12-055",
    "street": "Pomorska",
    "homeNumber": 12
  },
  "subjects": ["math", "physics", "materials"],
  "grades": [
    { "subjectName": "math", "subjectGrades": [5.0, 4.5, 4.0] },
    { "subjectName": "physics", "subjectGrades": [4.0, 4.5, 4.0] },
    { "subjectName": "materials", "subjectGrades": [3.0, 3.0, 3.0] }
  ]
})

```

```

db.student.insertOne({
  "indexNumber": 123123,
  "semester": 6,
  "faculty": "WIMIR",
  "firstName": "Kamil",
  "lastName": "Kowalski",
  "age": 19,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "32-055",
    "street": "Rynek",
    "homeNumber": 15
  },
  "subjects": ["Cprogramming", "DataModeling", "math"],
  "grades": [
    { "subjectName": "Cprogramming", "subjectGrades": [3.0, 2.5, 2.0] },
    { "subjectName": "DataModeling", "subjectGrades": [4.0, 4.5, 4.0] },
    { "subjectName": "math", "subjectGrades": [3.0, 3.0, 5.0] }
  ]
})

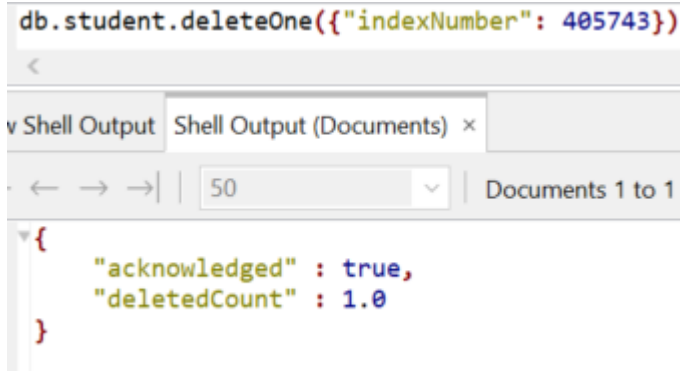
```

Usuwanie:

- a) Chcemy usunąć studenta o numerze indeksu 405743

```
db.student.deleteOne({ "indexNumber": 405743 })
```

```
db.student.deleteOne({ "indexNumber": 405743 })
```



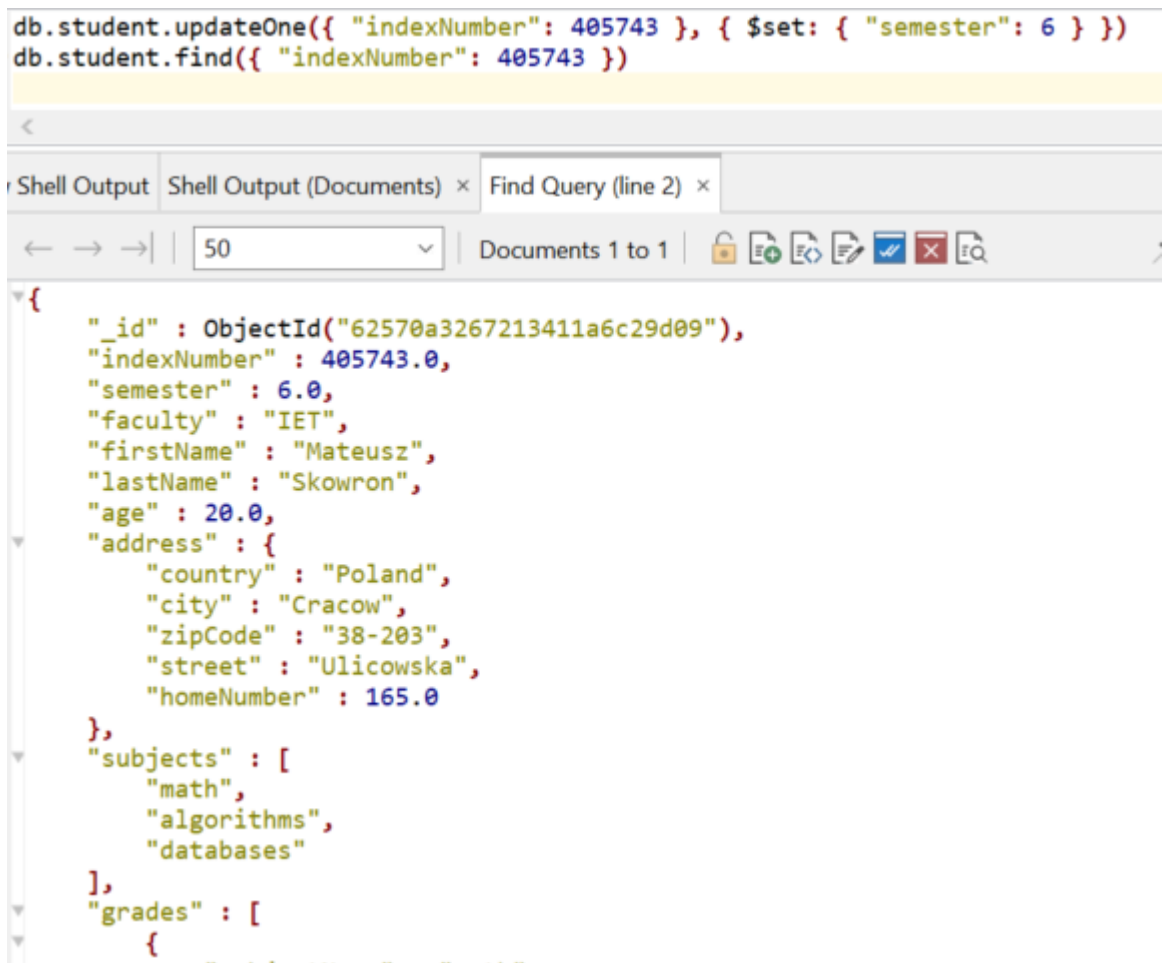
```
{
  "acknowledged" : true,
  "deletedCount" : 1.0
}
```

Modyfikacja:

- a) Chcemy zmienić semestr na 6 dla studenta o numerze indeksu 405743

```
db.student.updateOne({ "indexNumber": 405743 }, { $set: { "semester": 6 } })
db.student.find({ "indexNumber": 405743 })
```

```
db.student.updateOne({ "indexNumber": 405743 }, { $set: { "semester": 6 } })
db.student.find({ "indexNumber": 405743 })
```

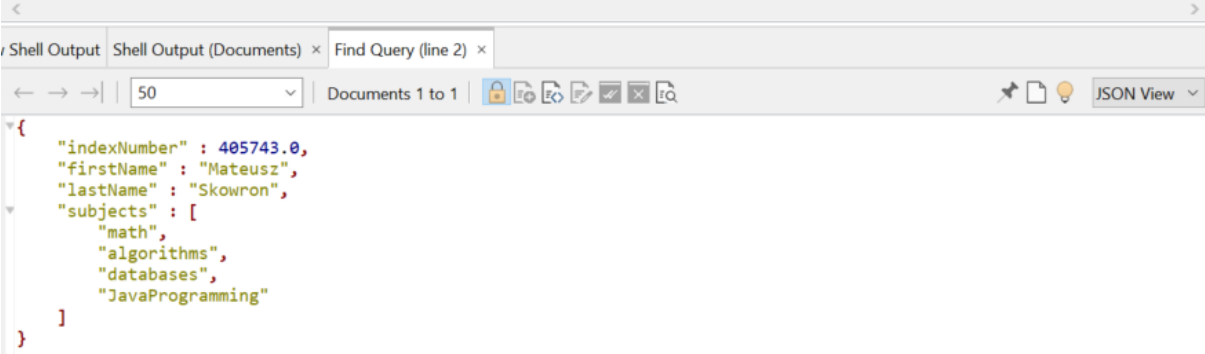


```
{
  "_id" : ObjectId("62570a3267213411a6c29d09"),
  "indexNumber" : 405743.0,
  "semester" : 6.0,
  "faculty" : "IET",
  "firstName" : "Mateusz",
  "lastName" : "Skowron",
  "age" : 20.0,
  "address" : {
    "country" : "Poland",
    "city" : "Cracow",
    "zipCode" : "38-203",
    "street" : "Ulicowska",
    "homeNumber" : 165.0
  },
  "subjects" : [
    "math",
    "algorithms",
    "databases"
  ],
  "grades" : [
    {
      ...
    }
  ]
}
```

b) Chcemy dodać przedmiot JavaProgramming dla studenta o indeksie 405743

```
db.student.updateOne({ "indexNumber": 405743 }, { $push: { subjects:
"JavaProgramming" } })
db.student.find({ "indexNumber": 405743 }, { "firstName": 1, "lastName": 1,
"indexNumber": 1, "subjects": 1, _id: 0 })
```

```
db.student.updateOne({ "indexNumber": 405743 }, { $push: { subjects: "JavaProgramming" } })
db.student.find({ "indexNumber": 405743 }, { "firstName": 1, "lastName": 1, "indexNumber": 1, "subjects": 1, _id: 0 })
```



```
{
  "indexNumber" : 405743.0,
  "firstName" : "Mateusz",
  "lastName" : "Skowron",
  "subjects" : [
    "math",
    "algorithms",
    "databases",
    "JavaProgramming"
  ]
}
```

c) Chcemy dodać do ocen przedmiot JavaProgramming, a z nim ocenę 3.0

```
db.student.updateOne({ "indexNumber": 405743 }, { $push: { "grades": {  
  "subjectName": "JavaProgramming", "subjectGrades": [3.0] } } })  
db.student.find({ "indexNumber": 405743 }, { "indexNumber": 1, "subjects": 1,  
  "grades": 1, _id: 0 })
```

```
db.student.updateOne({ "indexNumber": 405743 }, { $push: { "grades": { "subjectName": "JavaProgramming", "subjectGrades": [3.0] } } })  
db.student.find({ "indexNumber": 405743 }, { "indexNumber": 1, "subjects": 1, "grades": 1, _id: 0 })
```

Shell Output | Shell Output (Documents) x | Find Query (line 2) x

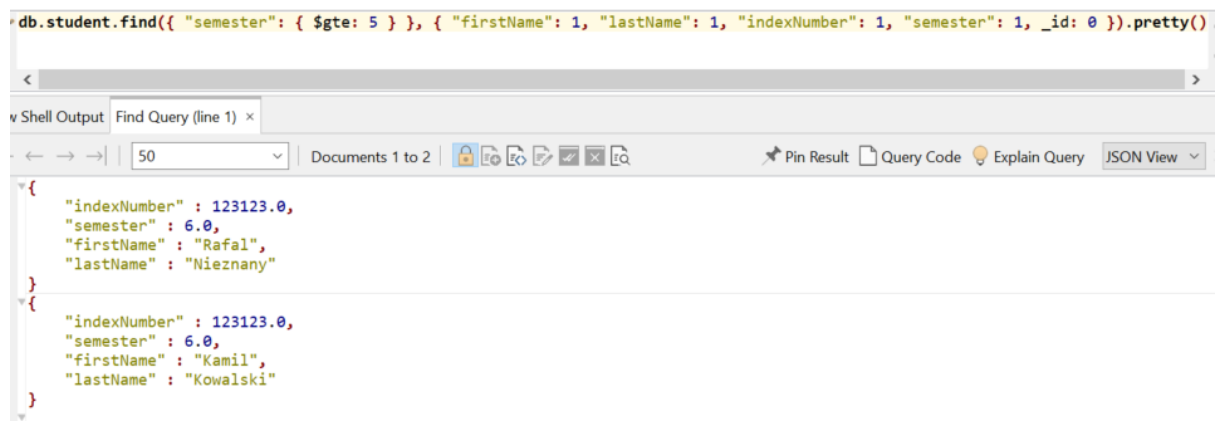
← → → | 50 | Documents 1 to 1 | 🔒 🔍 📄 📋 🗑️ 🔍 | 📌 Pin Result 📄 Query Code 💡 Explain Query JSON View

```
{  
  "indexNumber" : 405743.0,  
  "subjects" : [  
    "math",  
    "algorithms",  
    "databases"  
  ],  
  "grades" : [  
    {  
      "subjectName" : "math",  
      "subjectGrades" : [  
        5.0,  
        4.5,  
        4.0  
      ]  
    },  
    {  
      "subjectName" : "algorithms",  
      "subjectGrades" : [  
        3.0,  
        3.5,  
        3.0  
      ]  
    },  
    {  
      "subjectName" : "databases",  
      "subjectGrades" : [  
        5.0,  
        5.0,  
        5.0  
      ]  
    },  
    {  
      "subjectName" : "JavaProgramming",  
      "subjectGrades" : [  
        3.0  
      ]  
    }  
  ]  
}
```

Wyszukiwanie:

1. Chcemy znaleźć studentów, którzy są przynajmniej na 4 semestrze.

```
db.student.find({ "semester": { $gte: 5 } }, { "firstName": 1, "lastName": 1, "indexNumber": 1, "semester": 1, _id: 0 }).pretty()
```



The screenshot shows a MongoDB Shell window with the following query and results:

```
db.student.find({ "semester": { $gte: 5 } }, { "firstName": 1, "lastName": 1, "indexNumber": 1, "semester": 1, _id: 0 }).pretty()
```

The results are displayed in a JSON array:

```
[{"indexNumber": 123123.0, "semester": 6.0, "firstName": "Rafal", "lastName": "Nieznany"}, {"indexNumber": 123123.0, "semester": 6.0, "firstName": "Kamil", "lastName": "Kowalski"}]
```

2. Chcemy znaleźć studentów, mieszkających w Krakowie.

```
db.student.find({ "address.city": 'Cracow' }, { "firstName": 1, "lastName": 1, "address": 1, _id: 0 })
```



The screenshot shows a MongoDB Shell window with the following query and results:

```
db.student.find({ "address.city": 'Cracow' }, { "firstName": 1, "lastName": 1, "address": 1, _id: 0 })
```

The results are displayed in a JSON array:

```
[{"firstName": "Kamil", "lastName": "Kowalski", "address": {"country": "Poland", "city": "Cracow", "zipCode": "32-055", "street": "Rynek", "homeNumber": 15.0}}, {"firstName": "Mateusz", "lastName": "Skowron", "address": {"country": "Poland", "city": "Cracow", "zipCode": "38-203", "street": "Ulicowska", "homeNumber": 165.0}}]
```

3. Chcemy znaleźć studentów, którzy mają Programowanie w języku C na studiach.

```
db.student.find({ subjects: { $in: ["Cprogramming"] } }, { "firstName": 1, "lastName": 1, "subjects": 1, _id: 0 })
```

```
db.student.find({ subjects: { $in: ["Cprogramming"] } }, { "firstName": 1, "lastName": 1, "subjects": 1, _id: 0 })
```



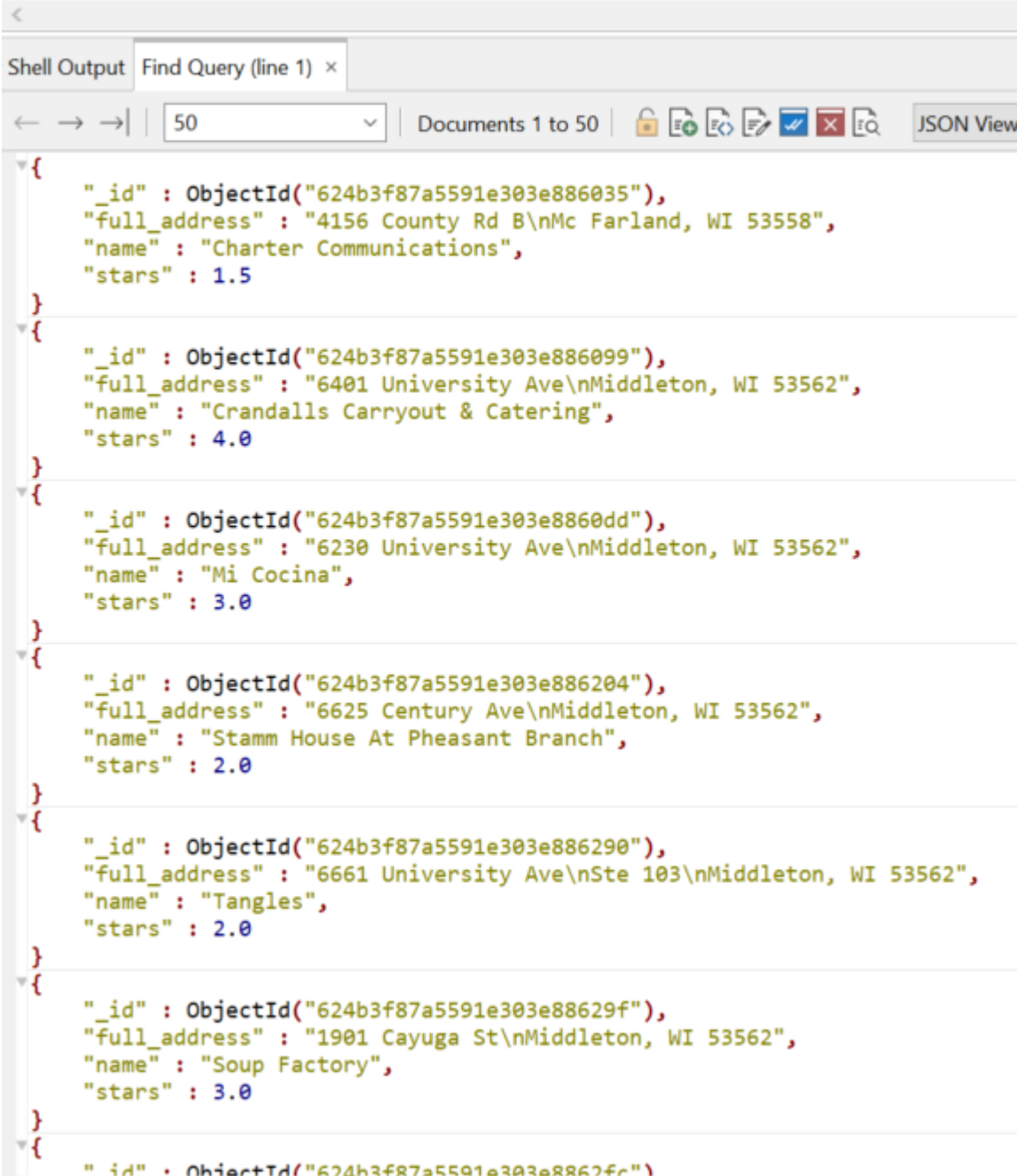
The screenshot shows a MongoDB Shell window with a query and its result. The query is: `db.student.find({ subjects: { $in: ["Cprogramming"] } }, { "firstName": 1, "lastName": 1, "subjects": 1, _id: 0 })`. The result is a JSON object: `{ "firstName": "Kamil", "lastName": "Kowalski", "subjects": ["Cprogramming", "DataModeling", "math"] }`. The interface includes a "Shell Output" tab, a "Find Query (line 1)" search bar, and a "Documents 1 to 1" indicator. There are also icons for "Pin Result", "Query Code", and "Explains".

Zadanie 8

a)

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

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



Shell Output Find Query (line 1) ×

← → → | 50 | Documents 1 to 50 | 🔒 📄 📄 📄 📄 📄 📄 📄 📄 📄 JSON View

```
{  
  "_id" : ObjectId("624b3f87a5591e303e886035"),  
  "full_address" : "4156 County Rd B\nMc Farland, WI 53558",  
  "name" : "Charter Communications",  
  "stars" : 1.5  
}  
{  
  "_id" : ObjectId("624b3f87a5591e303e886099"),  
  "full_address" : "6401 University Ave\nMiddleton, WI 53562",  
  "name" : "Crandalls Carryout & Catering",  
  "stars" : 4.0  
}  
{  
  "_id" : ObjectId("624b3f87a5591e303e8860dd"),  
  "full_address" : "6230 University Ave\nMiddleton, WI 53562",  
  "name" : "Mi Cocina",  
  "stars" : 3.0  
}  
{  
  "_id" : ObjectId("624b3f87a5591e303e886204"),  
  "full_address" : "6625 Century Ave\nMiddleton, WI 53562",  
  "name" : "Stamm House At Pheasant Branch",  
  "stars" : 2.0  
}  
{  
  "_id" : ObjectId("624b3f87a5591e303e886290"),  
  "full_address" : "6661 University Ave\nSte 103\nMiddleton, WI 53562",  
  "name" : "Tangles",  
  "stars" : 2.0  
}  
{  
  "_id" : ObjectId("624b3f87a5591e303e88629f"),  
  "full_address" : "1901 Cayuga St\nMiddleton, WI 53562",  
  "name" : "Soup Factory",  
  "stars" : 3.0  
}  
{  
  "_id" : ObjectId("624b3f87a5591e303e8862fc")
```

b)

```
db.business.find(  
  { "stars": { $eq: 5 } },  
)<div data-bbox="181 200 450 244" data-label="Text">

```
db.business.find(
 { "stars": { $eq: 5 } },
)<div data-bbox="177 279 281 295" data-label="Text">

Shell Output


```


```

5097

d)

```
db.business.distinct("city")
```

```
db.business.distinct("city")
```

<

Shell Output

Shell Output (Array) ×

← → →

50

Documents 1 to 1

```
[  
  "Ahwatukee",  
  "Anthem",  
  "Apache Junction",  
  "Arcadia",  
  "Atlanta",  
  "Avondale",  
  "Black Canyon City",  
  "Bonnyrigg",  
  "Boulder City",  
  "Buckeye",  
  "C Las Vegas",  
  "Cambridge",  
  "Carefree",  
  "Casa Grande",  
  "Cave Creek",  
  "Centennial Hills",  
  "Central City Village",  
  "Central Henderson",  
  "Chandler",  
  "Chandler-Gilbert",  
  "City of Edinburgh",  
  "Clark County",  
  "Columbus",  
  "Coolidge",  
  "Cottage Grove",  
  "Cramond",  
  "Dalkeith",  
  "Dane",  
  "De Forest",  
  "DeForest",  
  "Deforest",  
  "Eagan",  
  "Edinburgh",  
  "El Mirage",  
  "Enterprise",  
  "Fitchburg",  
  "Florence",  
  "Fort Kinnaird",  
  "Fort McDowell",  
]
```

c)

```
db.business.aggregate([
  { $match: { "categories": { $in: ["Restaurants"] } } },
  {
    $group:
    {
      _id: "$city",
      count: { $sum: 1 }
    }
  }
])
```

```
db.business.aggregate([
  { $match: { "categories": { $in: ["Restaurants"] } } },
  {
    $group:
    {
      _id: "$city",
      count: { $sum: 1 }
    }
  }
])
```

Shell Output | Aggregate Query (line 1) ×

← → | 50 | Documents 1 to 50 | 🔍 📄 💡

{
"_id" : "Gila Bend",
"count" : 9.0
}
{
"_id" : "N Las Vegas",
"count" : 7.0
}
{
"_id" : "Gold Canyon",
"count" : 15.0
}
{
"_id" : "Cave Creek",
"count" : 63.0
}
{
"_id" : "Casa Grande",
"count" : 61.0
}
{
"_id" : "Rio Verde",
"count" : 1.0
}
{
"_id" : "Cambridge",
"count" : 2.0
}
{
"_id" : "Peoria",
"count" : 221.0
}

g)

```
db.user.aggregate([
  { $match: { $or: [{ "votes.funny": { $eq: 0 } }, { "votes.useful": { $eq: 0 } } ] } },
  {
    $sort:
    {
      "name": 1
    }
  }
])
```

```
db.user.aggregate([
  { $match: { $or: [{ "votes.funny": { $eq: 0 } }, { "votes.useful": { $eq: 0 } } ] } },
  {
    $sort:
    {
      "name": 1
    }
  }
])
```

Shell Output Aggregate Query (line 1) ×

← → 50 Documents 1 to 50 [EQ] Pin Result Query Code Explain Query JSON View

```
1 {
2   "_id" : ObjectId("624b3fd2a5591e303ea12599"),
3   "yelping_since" : "2009-08",
4   "votes" : {
5     "funny" : NumberInt(0),
6     "useful" : NumberInt(0),
7     "cool" : NumberInt(0)
8   },
9   "review_count" : NumberInt(1),
10  "name" : " Bernard",
11  "user_id" : "xP3SPgfgW2vc5Zj5uV8SEA",
12  "friends" : [
13
14  ],
15  "fans" : NumberInt(0),
16  "average_stars" : 5.0,
17  "type" : "user",
18  "compliments" : {
19
20  },
21  "elite" : [
22
23  ]
24 }
25 {
26   "_id" : ObjectId("624b3fe1a5591e303ea652dd"),
27   "yelping_since" : "2011-12",
28   "votes" : {
29     "funny" : NumberInt(0),
30     "useful" : NumberInt(2),
31     "cool" : NumberInt(2)
32   },
33   "review_count" : NumberInt(10),
34   "name" : "Anastacia",
```

Zadanie 9 - A

1. Struktura

Po namyśle zdecydowałem, że baza będzie składać się z 3 kolekcji (wykładowcy, przedmioty, studenci). Wydaje mi się, że jest to optymalne rozwiązanie. Można by trzymać wszystko w jednej kolekcji, jednakże według mnie byłaby to zbędna redundancja danych, chociaż dostęp do poszczególnych danych patrząc na wykonywanie operacji na bazie byłby dużo prostszy, bez konieczności referencji do innej kolekcji.

Uważam, że takie rozgraniczenie na 3 kolekcje zwiększa czytelność bazy danych oraz ułatwia korzystanie z niej. Elementy są logicznie pogrupowane przez co wiemy co i gdzie mamy szukać. Jedyny minus tego rozwiązania to konieczność referencji do innych kolekcji (odwoływanie się po ID). Kolekcja wykładowcy oraz studenci jest oczywistym wyborem. Zdecydowałem się stworzyć osobną kolekcję przedmioty, choć mogłem w każdej z dwóch kolekcji trzymać potrzebne informacje o przedmiocie. Zdecydowałem się na takie rozwiązanie, gdyż informacji o przedmiocie jest sporo i nie każda z nich jest potrzebna w każdej z tych dwóch kolekcji, a może zdarzyć się operacja, która będzie wymagała więcej informacji, jednakże najczęściej wymagane są tylko niektóre dane, więc wybrałem najważniejsze informacje do każdej z nich oraz umieściłem referencję do kolekcji przedmioty w razie konieczności szczegółowych danych na temat przedmiotu.

Poniżej przedstawiam strukturę dokumentu każdej z kolekcji i krótki opis wybranego rozwiązania problemu.

a) Wykładowcy (lecturers)

Kolekcja przechowuje dane na temat prowadzących. W polu *subjects* umieszczone są dwie moim zdaniem najczęściej potrzebne informacje na ich temat oraz referencja do dokumentu kolekcji przedmioty, jeśli potrzebne byłby szczegółowe informacje.

Struktura dokumentu:

```
{
  "_id":      ObjectId(),
  "faculty":  [(faculty shortcut)],
  "degree":   [(degree name)],
  "firstName": (first name),
  "lastName": (last name),
  "age":      (age),
  "address": {
    "country": (country name),
    "city":    (city name),
    "zipCode": (zip-code like '38-203'),
    "street":  (street name),
    "homeNumber": (home number)
  },
  "subjects": [
    {
      "subjectName": (subject name),
      "role":        : (lecturer/coordinator),
      "subjectID"   : ObjectId()
    }
  ],
}
```

Przykładowy dokument:

```
{
  "_id" : ObjectId("5126bc054aed4daf9e2ab772"),
  "faculty": ["IET","EAIIB"],
  "degree" : ["dr","inż","prof"],
  "firstName": "Janusz",
  "lastName": "Profesorowski",
  "age": 45,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-203",
    "street": "Ulicowska",
    "homeNumber": 165
  },
  "subjects": [
    {"subjectName": "Cprogramming", "role": "lecturer", "subjectID": ObjectId("5126bc054aed4daf9e2ab772")},
    {"subjectName": "Algorithms", "role": "lecturer", "subjectID": ObjectId("5126bc054aed4daf9e2ab772")},
    {"subjectName": "C++programming", "role": "coordinator", "subjectID": ObjectId("5126bc054aed4daf9e2ab772")}
  ]
}
```

b) Studenci (students)

Kolekcja przechowuje dane na temat studentów. W polu *subjects* umieszczona jest jedynie nazwa przedmiotu (która moim zdaniem jest najczęściej potrzebna patrząc na tę kolekcję) i referencja jeśli wymagane są szczegółowe informacje o przedmiocie. Podobnie w przypadku pola *grades*.

Struktura dokumentu:

```
{
  "_id":      ObjectId(),
  "indexNumber": (index number),
  "semester": (semester number),
  "faculty":   (faculty shortcut),
  "degree":    [(degree name)],
  "firstName": (first name),
  "lastName":  (last name),
  "age":       (age),
  "address": {
    "country":  (country name),
    "city":     (city name),
    "zipCode":  (zip-code like '38-203'),
    "street":   (street name),
    "homeNumber": (home number)
  },
  "subjects": [
    {"subjectName": (subject name), "subjectID": ObjectId()},
  ],
  "grades": [
    {"subjectName": (subject name), "subjectID": ObjectId(), "subjectGrades": [(grade)]},
  ]
}
```

Przykładowy dokument:

```
{
  "_id" : ObjectId("5126bc054aed4daf9e2ab772"),
  "indexNumber": 405743,
  "semester": 4,
  "faculty": "IET",
  "firstName": "Mateusz",
  "lastName": "Skowron",
  "age": 20,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-203",
    "street": "Ulicowska",
    "homeNumber": 165
  },
  "subjects": [
    {"subjectName": "Math", "subjectID": ObjectId("5126bc054aed4daf9e2ab772")},
    {"subjectName": "Databases", "subjectID": ObjectId("5126bc054aed4daf9e2ab772")},
    {"subjectName": "Algorithms", "subjectID": ObjectId("5126bc054aed4daf9e2ab772")}
  ],
  "grades": [
    {"subjectName": "Math", "subjectID": ObjectId("5126bc054aed4daf9e2ab772"), "subjectGrades": [5.0,4.5,4.0]},
    {"subjectName": "Databases", "subjectID": ObjectId("5126bc054aed4daf9e2ab772"), "subjectGrades": [3.0,3.5,3.0]},
    {"subjectName": "Algorithms", "subjectID": ObjectId("5126bc054aed4daf9e2ab772"), "subjectGrades": [5.0,5.0,5.0]}
  ]
}
```

c) Przedmioty (subjects)

Kolekcja przechowuje szczegółowe dane na temat przedmiotów. W polu *reviews* (oceny przedmiotu) znajduje się referencja do konkretnego studenta, jednak najistotniejsze jest jego imię nazwisko oraz sama treść. Jeśli wymagane są dodatkowe informacje wtedy należy odwołać się do kolekcji studentów poprzez tę referencję. Oceny studentów można by przechowywać w kolekcji *students* jednak logicznej jest mieć tę informację w dokumencie reprezentującym dany przedmiot.

Struktura dokumentu:

```
{
  "_id":          ObjectId(),
  "name":         (subject name),
  "fieldOfStudy": (field of study),
  "mandatory":    (obligatory/elective),
  "lectureLanguage": (language),
  "faculty":      (faculty shortcut),
  "formOfVerification": (exam/assessment),
  "numberOfHours": {
    "noLectures":      (number of lectures),
    "auditoriumClasses": (number of auditorium classes),
    "laboratoryClasses": (number of laboratory classes),
  },
  "coordinator": { "name": (first name and last name), "id": ObjectId() },
  "lecturers":   [ { "name": (first name and last name), "id": ObjectId() } ],
  "literature":  [ (book info) ],
  "ECTSPoints":  (number of ECTS points),
  "goals":       (goal of the subject),
  "studyContent" : (content description),
  "additionalInfo": (some additional informations if needed)
  "reviews": [ { "studentID": ObjectId(), "studentName": (first name and last name), "content": (review content) } ]
}
```

Przykładowy dokument:

```
{
  "_id" : ObjectId(),
  "name" : "Cprogramming",
  "fieldOfStudy": "Computer Science",
  "mandatory": "obligatory",
  "lectureLanguage": "Polish",
  "faculty": "IET",
  "formOfVerification": "exam",
  "numberOfHours": {
    "noLectures": 28,
    "auditoriumClasses": 36,
    "laboratoryClasses": 30,
  },
  "coordinator": { "name": "Mateusz Skowron", "id": ObjectId() },
  "lecturers": [ { "name": "Mateusz Skowron", "id": ObjectId() } ],
  "literature": [ "J. Grębosz, Opus magnum C++11", "J. Grębosz, Symfonia C++ Standard", "B. Stroustrup, Język C++" ],
  "ECTSPoints": "5.0",
  "goals": "Learn student how to code in C language.",
  "studyContent": "Classes, constructors. OOP and many many more.",
  "additionalInfo": "Lectures are online on Webex platform.",
  "reviews": [
    {
      "studentID": ObjectId(),
      "studentName": "Jan kowalski",
      "content": "Bardzo wartościowy przedmiot z najlepszym prowadzącym. Egzamin to formalność."
    }
  ]
}
```


2. Baza danych

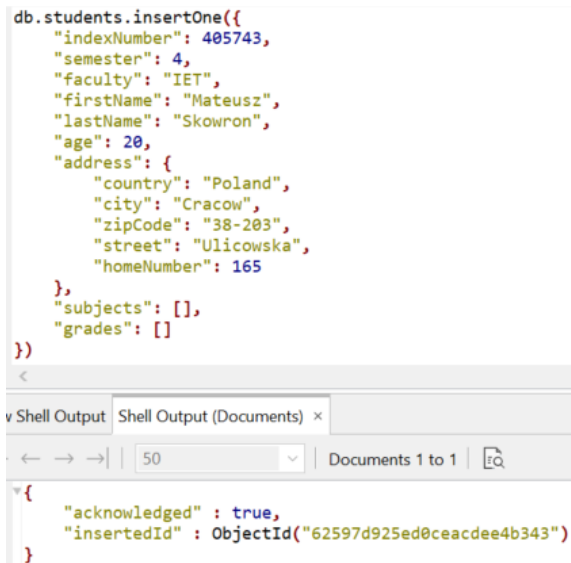
a) Stworzenie bazy danych oraz potrzebnych kolekcji.

```
test> use zad9DB_MS
switched to db zad9DB_MS
zad9DB_MS> db.createCollection("students")
{ ok: 1 }
zad9DB_MS> db.createCollection("lecturers")
{ ok: 1 }
zad9DB_MS> db.createCollection("subjects")
{ ok: 1 }
zad9DB_MS>
```

b) Wprowadzenie danych

Wprowadziłem kilka przykładowych przedmiotów, studentów oraz wykładowców. Poniżej widoczny jest przykładowych poleceń *insertOne()* oraz *find()* na każdej z nich.

InsertOne():



```
db.students.insertOne({
  "indexNumber": 405743,
  "semester": 4,
  "faculty": "IET",
  "firstName": "Mateusz",
  "lastName": "Skowron",
  "age": 20,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-203",
    "street": "Ulicowska",
    "homeNumber": 165
  },
  "subjects": [],
  "grades": []
})
```

Shell Output (Documents) ×

← → | 50 | Documents 1 to 1 | EQ

Pin Result JSON View

```
{
  "acknowledged" : true,
  "insertedId" : ObjectId("62597d925ed0ceacdee4b343")
}
```

```

db.lecturers.insertOne({
  "faculty": ["IET", "EAIIB"],
  "degree": ["dr", "inż", "prof"],
  "firstName": "Janusz",
  "lastName": "Profesorowski",
  "age": 45,
  "address": {
    "country": "Poland",
    "city": "Cracow",
    "zipCode": "38-223",
    "street": "Zamkowska",
    "homeNumber": 12
  },
  "subjects": []
})

```

w Shell Output

Shell Output (Documents) x

← → → |

50

Documents 1 to 1



Pin Result

JSON View



```

{
  "acknowledged": true,
  "insertedId": ObjectId("62597f355ed0ceacdee4b349")
}

```

```

db.subjects.insertOne({
  "name": "Databases",
  "fieldOfStudy": "Data bases",
  "mandatory": "elective",
  "lectureLanguage": "Polish",
  "faculty": "IET",
  "formOfVerification": "assessment",
  "numberOfHours": {
    "noLectures": 15,
    "auditoriumClasses": 0,
    "laboratoryClasses": 10,
  },
  "coordinator": {},
  "lecturers": [],
  "literature": ["Podstawy baz danych"],
  "ECTSPoints": "5.0",
  "goals": "Learn student how to code in C language.",
  "studyContent": "Classes, constructors. OOP and many many more.",
  "additionalInfo": "Lectures are online on Webex platform.",
  "reviews": []
})

```

w Shell Output

Shell Output (Documents) x

← → → |

50

Documents 1 to 1



Pin Result

JSON View



```

{
  "acknowledged": true,
  "insertedId": ObjectId("625998976d05c584d4bd95a")
}

```

find():

```
db.students.find({}, { "firstName": 1, "lastName": 1, "indexNumber": 1 })
```

Shell Output Find Query (line 1) ×

50 Documents 1 to 5

```
{
  "_id" : ObjectId("62597d925ed0ceacdee4b343"),
  "indexNumber" : 405743.0,
  "firstName" : "Mateusz",
  "lastName" : "Skowron"
}
{
  "_id" : ObjectId("62597e1b5ed0ceacdee4b344"),
  "indexNumber" : 123123.0,
  "firstName" : "Rafał",
  "lastName" : "Niezmany"
}
{
  "_id" : ObjectId("62597e465ed0ceacdee4b346"),
  "indexNumber" : 123123.0,
  "firstName" : "Kamil",
  "lastName" : "Kowalski"
}
{
  "_id" : ObjectId("62597e975ed0ceacdee4b347"),
  "indexNumber" : 405777.0,
  "firstName" : "Jurek",
  "lastName" : "Jurkowski"
}
{
  "_id" : ObjectId("62597ecc5ed0ceacdee4b348"),
  "indexNumber" : 432372.0,
  "firstName" : "Mariola",
  "lastName" : "Nowakowska"
}
```

```
db.lecturers.find({}, { "firstName": 1, "lastName": 1, "faculty": 1, "_id": 0 })
```

Shell Output Find Query (line 1) ×

50 Documents 1 to 5

```
{
  "faculty" : [
    "IET",
    "EAIIB"
  ],
  "firstName" : "Janusz",
  "lastName" : "Profesorowski"
}
{
  "faculty" : [
    "IET"
  ],
  "firstName" : "Mariusz",
  "lastName" : "Doktorski"
}
{
  "faculty" : [
    "IET"
  ],
  "firstName" : "Juliusz",
  "lastName" : "Slowacki"
}
{
  "faculty" : [
    "EAIIB"
  ],
  "firstName" : "Mateusz",
  "lastName" : "Piatkowski"
}
{
  "faculty" : [
    "WIMIR"
  ],
  "firstName" : "Adam",
  "lastName" : "Mickiewicz"
}
```

```
db.subjects.find({}, { "name": 1, "ECTSPoints": 1 })
```

Shell Output Find Query (line 1) ×

50 Documents 1 to 3

```
{
  "_id" : ObjectId("625983db5ed0ceacdee4b350"),
  "name" : "Cprogramming",
  "ECTSPoints" : "5.0"
}
{
  "_id" : ObjectId("625985065ed0ceacdee4b351"),
  "name" : "Algebra",
  "ECTSPoints" : "8.0"
}
{
  "_id" : ObjectId("625986885ed0ceacdee4b352"),
  "name" : "Databases",
  "ECTSPoints" : "2.0"
}
```

3. Operacje

a) Zapis studenta na przedmiot.

Przykładowo student został zapisany na przedmiot Cprogramming.

```
db.students.updateOne
(
  {
    "indexNumber": 405743 },
  {
    $push: {
      "subjects":
        {
          "subjectName": "Cprogramming",
          "subjectID": ObjectId("625983db5ed0ceacdee4b350")
        }
    }
  }
)

db.students.updateOne
(
  {
    "indexNumber": 405743 },
  {
    $push: {
      "grades":
        {
          "subjectName": "Cprogramming",
          "subjectID": ObjectId("625983db5ed0ceacdee4b350"),
          "subjectGrades": []
        }
    }
  }
)
)
```

Shell Output Shell Output (Documents) ×

50 Documents 1 to 2

```
{
  "acknowledged" : true,
  "matchedCount" : 1.0,
  "modifiedCount" : 1.0
}
{
  "acknowledged" : true,
  "matchedCount" : 1.0,
  "modifiedCount" : 1.0
}
```

b) Przyznanie koordynowania przedmiotu danemu prowadzącemu.

Przykładowo Juliusz Słowacki będzie prowadził przedmiot Cprogramming.

```
db.subjects.update(
  { "name": "Cprogramming", "faculty": "IET" },
  { $set: { "coordinator": { "name": "Juliusz Słowacki", "id": ObjectId("62597faf5ed0ceacdee4b34b") } } }
)

db.lecturers.update(
  { "_id": ObjectId("62597faf5ed0ceacdee4b34b") },
  {
    $push:
    {
      "subjects":
      {
        "subjectName": "Cprogramming",
        "role": "coordinator",
        "subjectID": ObjectId("625983db5ed0ceacdee4b350")
      }
    }
  }
)

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

c) Przyznanie prowadzenia zajęć danemu prowadzącemu.

Przykładowo Mariusz Doktorski będzie prowadził zajęcia z Cprogramming.

```
db.subjects.update(
  { "name": "Cprogramming", "faculty": "IET" },
  { $push: { "lecturers": { "name": "Mariusz Doktorski", "id": ObjectId("62597f7f5ed0ceacdee4b34a") } } }
)

db.lecturers.update(
  { "_id": ObjectId("62597f7f5ed0ceacdee4b34a") },
  {
    $push:
    {
      "subjects":
      {
        "subjectName": "Cprogramming",
        "role": "lecturer",
        "subjectID": ObjectId("625983db5ed0ceacdee4b350")
      }
    }
  }
)

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

d) Ocena przedmiotu przez studenta.

Przykładowo student ocenia przedmiot Cprogramming.

```
db.subjects.updateOne
(
  {
    "name": "Cprogramming", "faculty": "IET" },
  {
    $push: {
      "reviews":
        {
          "studentID": ObjectId("62597d925ed0ceacdee4b343"),
          "studentName": "Mateusz Skowron",
          "content": "Genialny przedmiot, mozna sie bardzo duzo nauczyc."
        }
    }
  }
)
```

Shell Output (Documents) ×

← → | 50 | Documents 1 to 1 | 🔍

Pin Result JSON View ⚙️

```
{
  "acknowledged" : true,
  "matchedCount" : 1.0,
  "modifiedCount" : 1.0
}
```

e) Wykładowca wystawia ocenę.

Przykładowo prowadzący wystawia ocenę studentowi o indeksie 405743 z przedmiotu Cprogramming.

```
db.students.update(
  { "grades.subjectName": "Cprogramming", "indexNumber": 405743 },
  { $push: { "grades.$.subjectGrades": 4.0 } }
)
```

Shell Output

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```