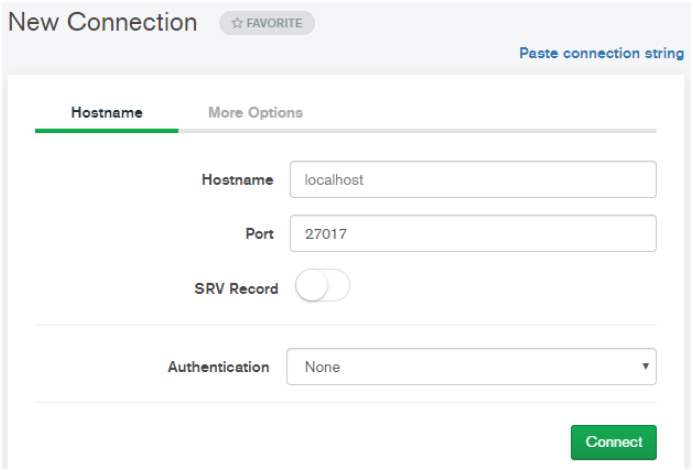
Създаване на бази данни и генериране на заявки чрез приложение Compass

Необходимо програмно осигуряване:

MongoDB сървър и приложението Compass.

Стартирайте локално сървъра MongoDB:

Стартирайте приложението Compass и реализирайте комуникация с локалния MongoDB сървър:



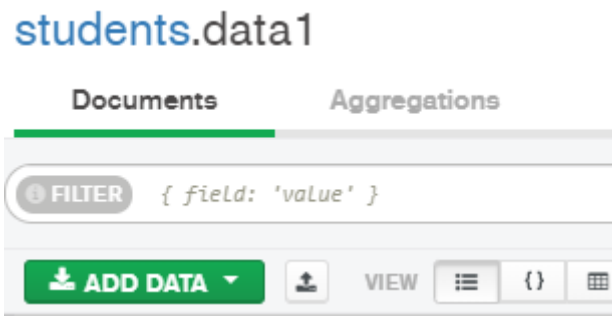
Задача 1: *Трябва да проектирате база данни, която описва студенти чрез следните свойства: идентификатор на студента (факултетен номер), име и фамилия, както и оценките по дисциплините от последния семестър. Да се има предвид, че трябва да могат да се правят чести заявки, свързани с оценките на студентите.*

Тъй като по условие на задачата трябва да се правят чести заявки, свързани с оценките на студентите през един семестър, то тази информация трябва да се вгради в документа, описващ студент. Следва примерна структура на документа:

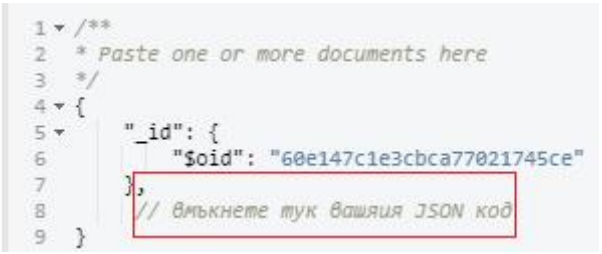
```
{
   "id": "21705001",
   "name": {
       "firstName": "Георги",
       "lastName": "Котев"
   },
   "grade": [
       {
           "subject": "НБД", "value": 4
       },
       {
           "subject": "КМС", "value": 6
       },
       {
           "subject": "ММС", "value": 5
       },
       {
           "subject": "ДСП", "value": 2
       }
   ]
}
```

Ще работим с графичния интерфейс, който MongoDB Compass предоставя. Създайте база данни “students”. Чрез бутон <+> създайте нова колекция с име “data1”:

След като създадете колекцията трябва да я населите с данни. Това се реализира чрез бутон <ADD DATA>:

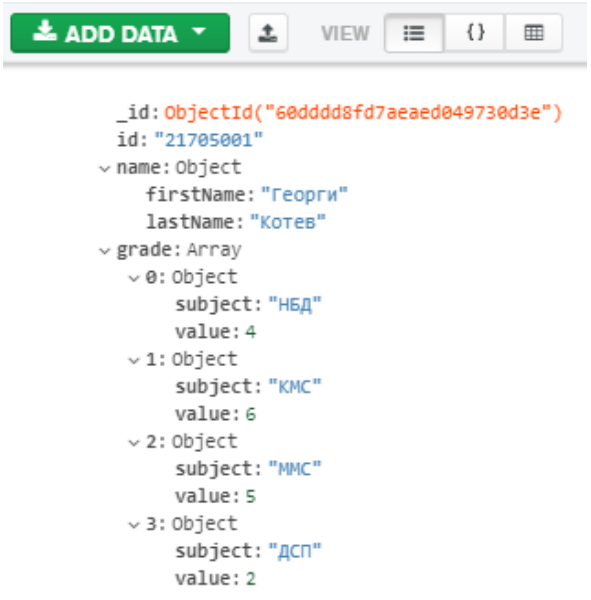


Ако имате съдържанието на документите в JSON или CSV файл, можете да ги импортирате чрез команда “Import File”. На този етап ще въведем три документа чрез редактора на Compass. За целта изберете вмъкване на документ – “Insert Document”. Редакторът ще генерира нов документ за вашата колекция в който има само поле “_id”.



```
/**
 * Paste one or more documents here
 */
{
    "_id": {
        "$oid": "60e147c1e3cbca77021745ce"
    },
    // вмъкнете тук вашия JSON код
}
```

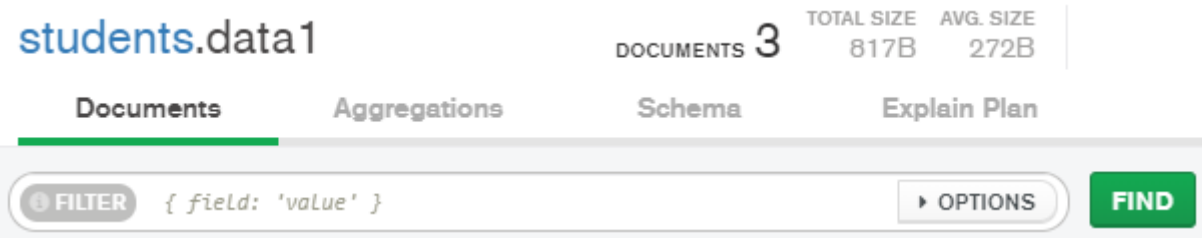
Веднага след него (не забравяйте символ запетая, който разделя полетата) въведете останалото съдържание на документа. След като въведете съдържанието на текущия документа, натиснете бутон <INSERT>. По аналогичен начин въведете още поне два документа в колекция “data1”:



Задача 2: *Трябва да генерирате заявка към база данни "students", колекция "data1" която да връща имената на всички студенти, които имат:*

- *Отлична оценка по дисциплината НБД.*
- *Оценка в интервала [4, 6] по дисциплината НБД.*

Приложението Compass позволява генерирането на произволни по сложност заявки към документите от избрана колекция. За целта изберете колекция "students.data1", а от хоризонталното меню - "Documents":



Ще използваме заявка с филтриране на резултата, за да изпълним задачата. За целта правилото за филтриране трябва да се въведе в поле "FILTER". Имате възможност да използвате операторите на MongoDB, свързани с филтриране. Те започват със знака за долар, например: $eq, $ne, $gt, $gte, $lt, $lte, $or, $nor, $and, $not, $exists, $in, $all, $regex, $elemMatch. Използвайте документацията на MongoDB и опишете предназначението на всеки един от тези оператори.

MongoDB оператор	Предназначение	Пример за използване на оператора
$eq		
$ne		
$gt		
$gte		
$lt		
$lte		
$or		
$nor		
$and		
$not		
$exists		
$in		
$all		
$regex		
$elemMatch		

Базовият формат на филтъра е следния:

```
{"field": "value"}
```

Това означава, че полето с име “field” трябва да има стойност “value”.

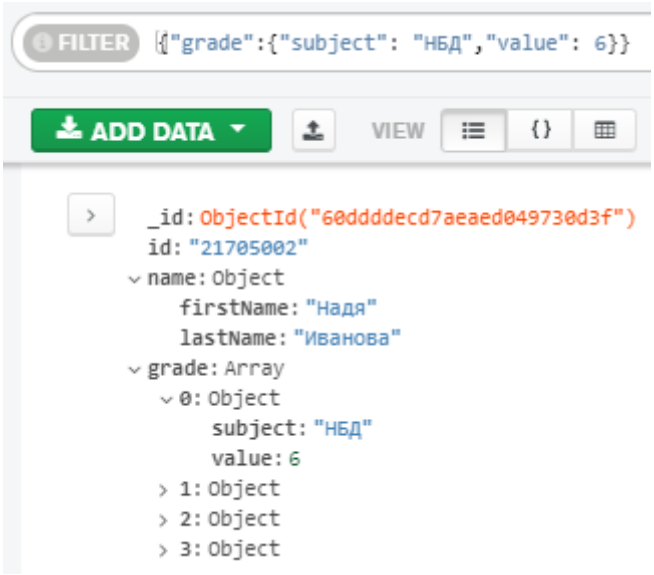
Нека да разгледаме ***първото условие на задачата*** - да намерим всички студенти, които имат оценка 6 по дисциплината НБД. В този случай трябва да филтрираме стойностите на две полета от масив “grade”: име на дисциплината – поле “subject” и стойността на оценката – поле “value”. Нека да пробваме със следния филтър:

```
{"grade.subject":" НБД ", "grade.value":6}
```

След изпълнение на заявката ще получим списък на студентите, които са изучавали дисциплина “НБД” и имат отлична оценка, независимо по коя дисциплина. Следователно, тази заявка не изпълнява условието на задачата. Трябва да зададем, че търсим специфична двойка стойности за дисциплина и оценка, например:

```
{"grade":{"subject": "НБД","value": 6}}
```

Тази заявка ще върне всички документи, за които двойката “subject-value” има зададените чрез филтъра стойности.



Ако е необходимо да получим само имената на студентите, а не целия документ, ще филтрираме част от полетата от резултата (документа), който MongoDB връща. За целта натиснете бутон “OPTIONS” и въведете следния низ в поле “PROJECT”:

```
{"_id":0, "name":1}
```

Всяко поле, което има стойност 1 ще бъде включено в резултата. Тъй като системното поле “_id” по подразбиране се включва във всеки резултат, за да го премахнем задаваме стойност 0.

Нека да реализираме *<u>второто условие на задачата</u>*. За да получим студентите, които имат оценки между 3 и 5 по дисциплината НБД ще използваме следния филтър:

{"grade":{ "subject": "НБД" , "value": { $gte: 3, $lte: 5 }}}

Интервалът за оценката е зададен с оператори $gte (≥) и $lte (≤). Въпреки, че синтаксисът изглежда логичен, той <u>няма да върне желания резултат</u>. Причината за това е, че тук имаме заявка с няколко (две) условия, приложена за вложени полета. В този случай трябва да се използва оператор $elemMatch, за да зададете множество критерии за масив от вградени документи, така че поне един вграден документ да отговаря на всички зададени критерии. Следователно филтърът трябва да бъде:

{"grade":{$elemMatch: { "subject": "НБД", "value": { $gte: 3, $lte: 5 } }}}

Задача 3: *Трябва да генерирате заявка към база данни "students", колекция "data1" която да връща имената на всички студенти, които имат оценка със стойност a или b по <u>поне една</u> от дисциплините НБД и КМС.*

Използвайте оператор $in чрез който да зададете от кои дисциплини се интересувате, както и кои оценки представляват интерес за вас:

{"grade":{$elemMatch:{"subject":{$in:["НБД","КМС"]}, "value": {$in:[5, 6] }}}}

Задача 4: *Трябва да генерирате заявка към база данни "students", колекция "data1" която да връща имената на всички студенти, които имат оценка по-висока от 2 за дисциплината НБД и оценка 4 или 5 за дисциплината КМС.*

Трябва да използвате два пъти оператор $elemMatch и да обедините тези условия чрез оператор $all:

```
{
    "grade": {
      $all:
         [
            {
              $elemMatch:{"subject":"НБД", "value": {$gt:2}}
            },
            {
              $elemMatch:{"subject":"КМС", "value": {$in:[4,5]}}
            }
         ]
    }
}
```

Задача 5: *Оптимизирайте схемата на база данни "students", колекция "data1" така, че да се намали размера на колекцията и да се повиши бързодействието на заявките.*

Ако разгледаме съдържанието на документ от колекция "students.data1" ще забележим, че описанието на оценките не е оптимално. Всяка дисциплина се описва от поле "subject", а името на дисциплината е стойността на това поле. Можем да зададем името на дисциплините да е ключ, а не стойност. По този начин размерът на колекцията ще намалее, а заявките, свързани с име на дисциплина ще се опростят и ще връщат резултат по бързо. Следва новото съдържание на документа:

```
{
    "id": "21705001",
    "name": {
      "firstName": "Георги",
      "lastName": "Котев"
    },
    "grade": {
      "НБД": { "value": 4 },
      "КМС": { "value": 6 },
      "ММС": { "value": 5 },
      "ДСП": { "value": 2 }
    }
}
```

Създайте нова колекция с име "data2" в базата данни "students" като спазвате новата схема. В Табл.1 са дадени заявките за всяка една от предложените схеми. Вижда се, че заявките към документите от колекция "data2" са много по-кратки и разбираеми. Освен това те се изпълняват и по-бързо.

Табл.1 Сравнение на заявките при използване на схемата от колекция “data1” и “data2”

Колекция “data1”	Колекция “data2”
{"grade":{"subject": "НБД","value": 6}}	{"grade.НБД.value":6}
{"grade":{$elemMatch:{"subject":"НБД","value":{$gte:3,$lte:5}}}}	{"grade.НБД.value":{$gte: 3, $lte: 5}}
{"grade":{$elemMatch:{"subject":{$in:["НБД","КМС"]}, "value": {$in:[5,6] }}}}	{$or:[{"grade.НБД.value": {$in:[5,6]}},{"grade.КМС.value": {$in:[5,6]}}]}
{ "grade": { $all: [{ $elemMatch:{"subject":"НБД", "value": {$gt:2}} }, { $elemMatch:{"subject":"КМС", "value": {$in:[4,5]}} }] } }	{$and:[{"grade.НБД.value":{$gt:2}},{"grade.КМС.value": {$in:[4,5]}}]}