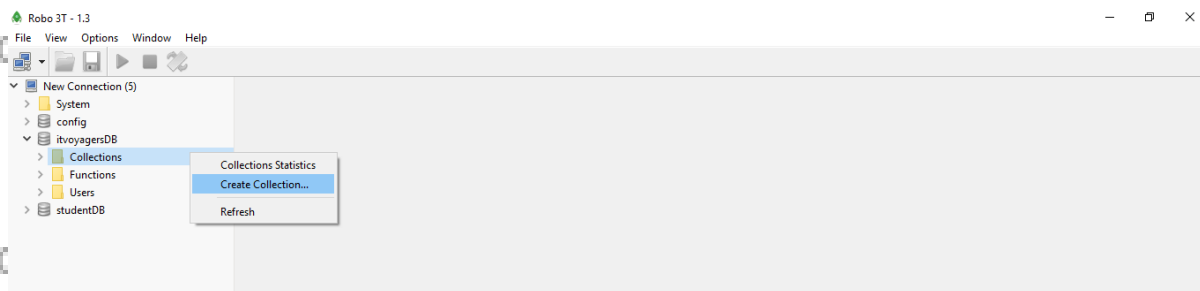


# Implementing Aggregation

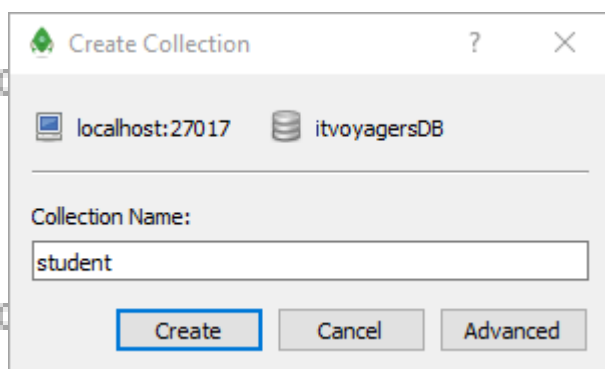
Let's create simple collection for student.

To understand the process of Robo 3T installation and connection process follow the document from this [link](#)

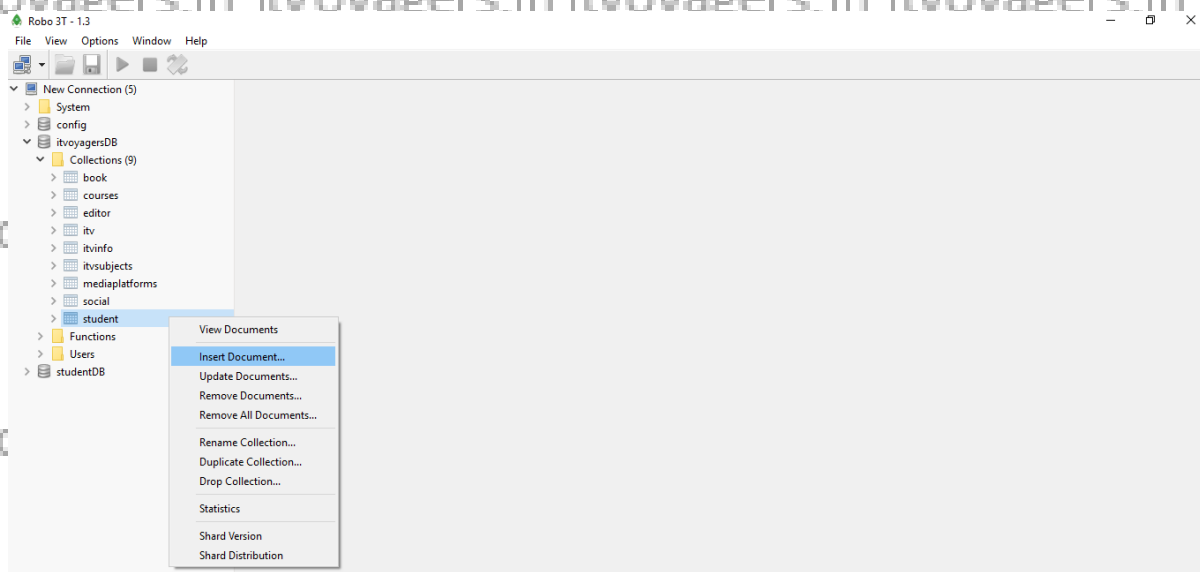
Let's us create new collection for this practical. Right click on **"Collections"** and select **"Create Collection..."**



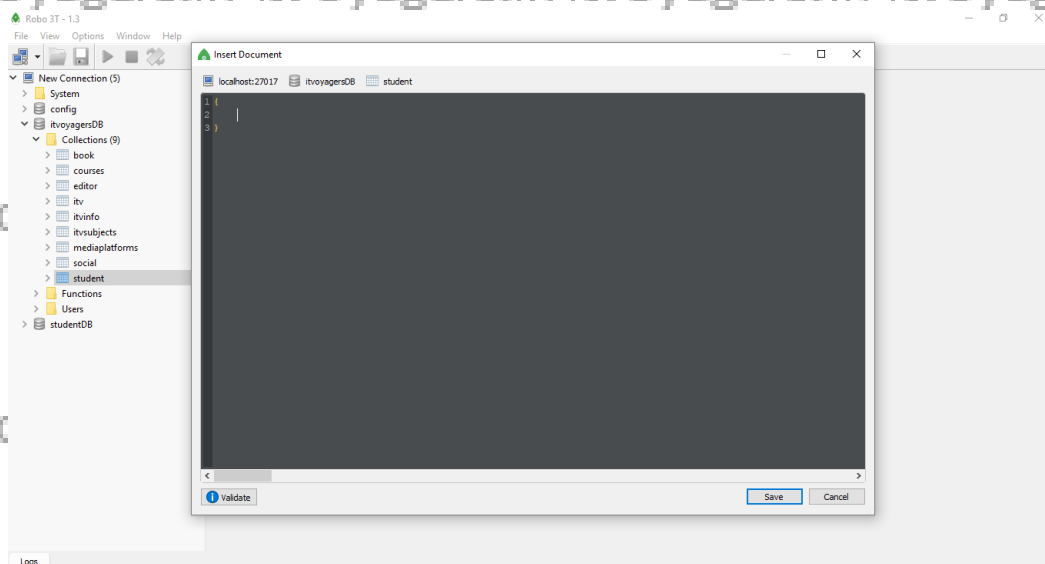
Type a name for the collection.



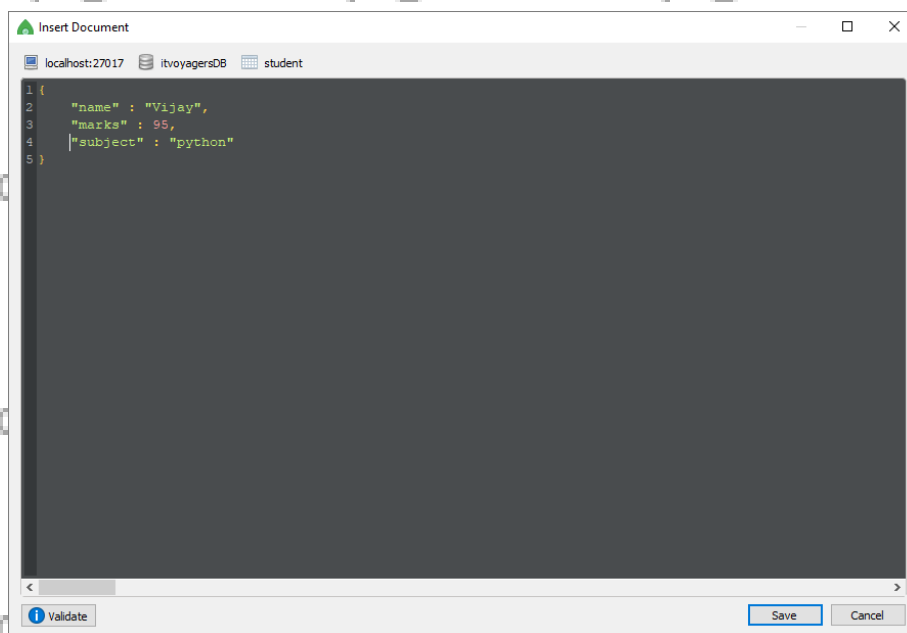
Click on **"Create"**. Once collection is created now let's insert few documents in this collection.



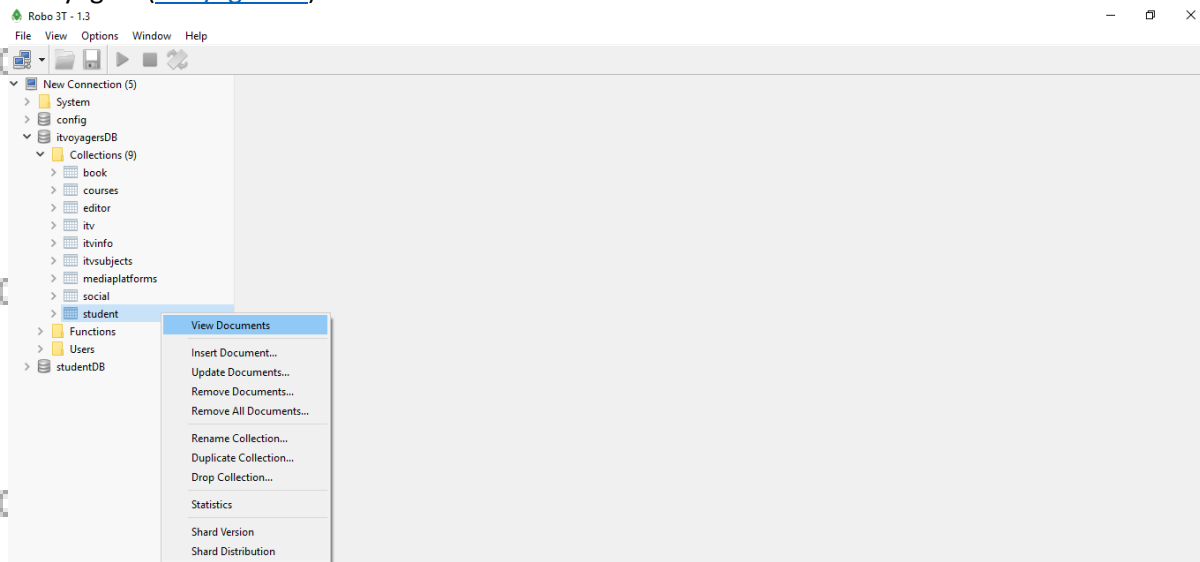
## Type document in this window



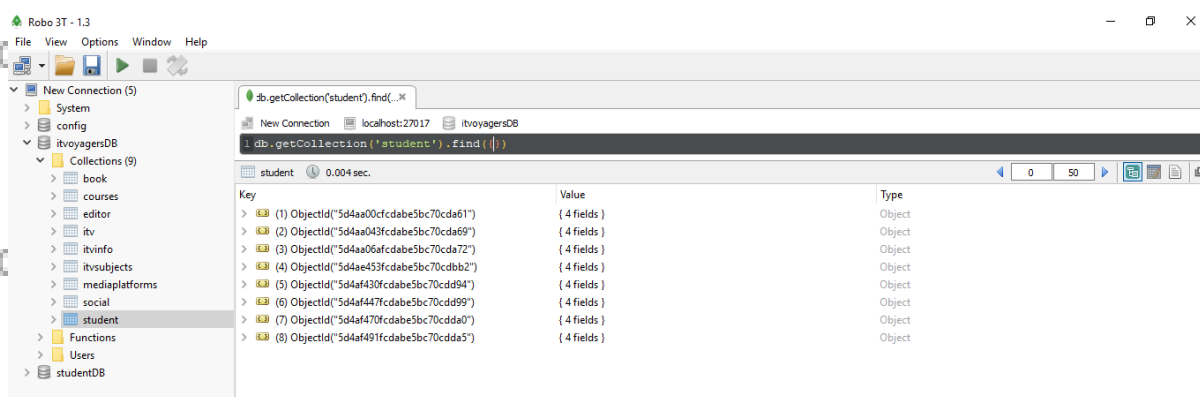
Following is our first document.



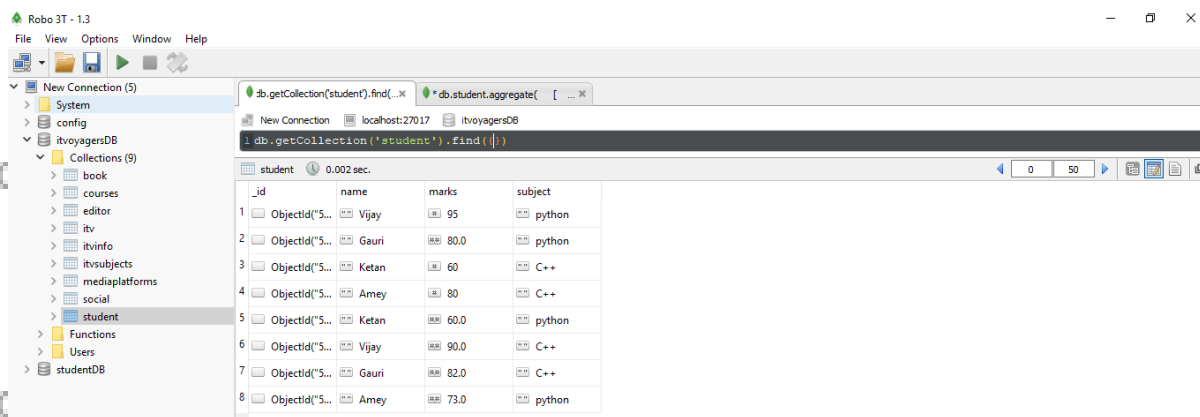
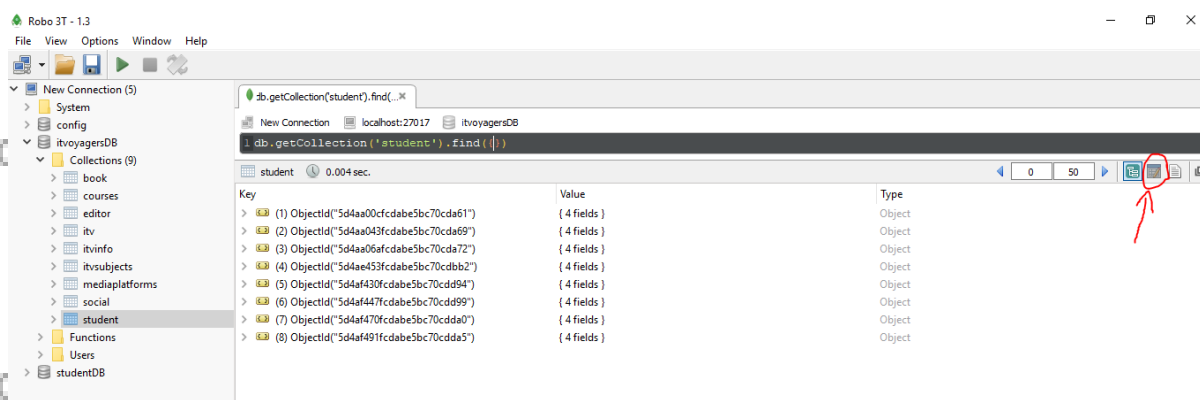
Once the documents is written click on “Save” and few more document using same process. Let’s view document.



We have inserted total 8 documents in collection.



To view all documents from the collection in different and more clear view click on **“View result in table mode”** button.



Now let's start performing following functions.

## Write a MongoDB query to use sum, avg, min and max expression.

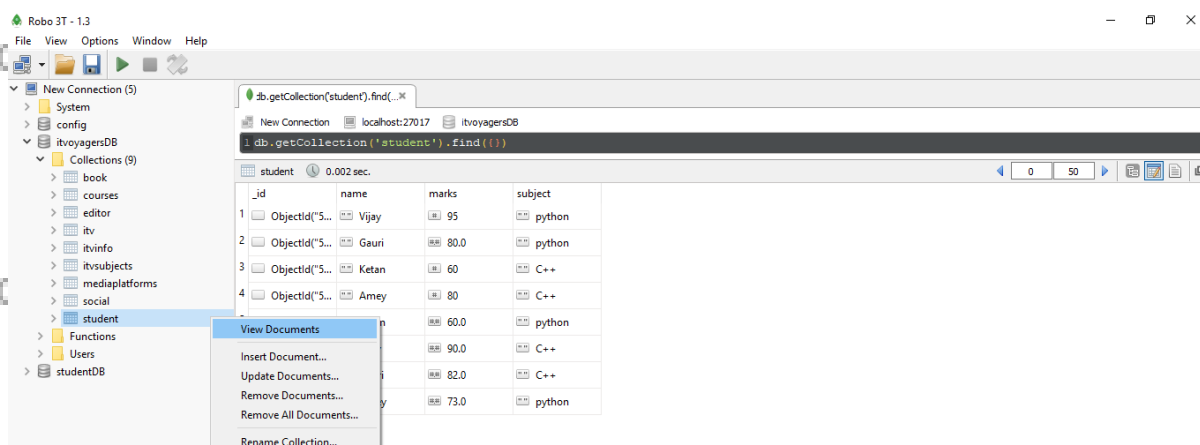
In mongoDB aggregate method we will be using \$match, \$group and \$sort as it parameter.

**\$match** : Allow us to set the condition on basis of which mongoDB will filter the data.

**\$group** : It has “\_id” and “total” by using which we can perform aggregate operations.

**\$sort** : It is use to order the data in ascending or descending order.

Let's start



### 1. sum

Suppose we need to add marks of both the subjects for each student in collection.

So we will set **\_id**: “\$name” which states that we will be grouping using “name” from each document.

And **total**: { \$sum : “\$marks” } which state that we need sum of all “marks” which has same “name” value.

New Connection localhost:27017 itvoyagersDB

```
1 db.student.aggregate(
2   [
3     { $match : { } },
4     { $group : { _id : "$name" , total : { $sum : "$marks" } } }
5   ]
6 )
7
```

**Output:**

	student	0.003 sec.
	_id	total
1	Amey	153.0
2	Gauri	162.0
3	Ketan	120.0
4	Vijay	185.0

**2. avg**

Suppose we need average marks of both the subjects for each student in collection.

So we will set **\_id: "\$name"** which states that we will be grouping using "name" from each document.

And **total : { \$avg : "\$marks" }** which state that we need average of all "marks" which has same "name" value.

```
New Connection localhost:27017 itvoyagersDB
1 db.student.aggregate (
2   [
3     { $match : { } },
4     { $group : { _id : "$name" , total : { $avg : "$marks" } } }
5   ]
6 )
7
```

**Output:**

	student	0.006 sec.
	_id	total
1	Amey	76.5
2	Gauri	81.0
3	Ketan	60.0
4	Vijay	92.5

If we want result to be displayed in descending order, we have to add **\$sort** in our query in **\$sort** we have we have to set **"total"** to -1.

**-1** represents descending order.

New Connection localhost:27017 itvoyagersDB

```
1 db.student.aggregate(  
2   [  
3     {$match : { } },  
4     {$group : { _id : "$name" , total : { $avg : "$marks" } } },  
5     {$sort : {total : -1} }  
6   ]  
7 )
```

**Output:**

student 0.003 sec.	
_id	total
1 "Vijay"	92.5
2 "Gauri"	81.0
3 "Amey"	76.5
4 "Ketan"	60.0

If we want result to be displayed in Ascending order, we have to add **\$sort** in our query in **\$sort** we have to set **"total"** to 1.

**1** represents descending order.

New Connection localhost:27017 itvoyagersDB

```
1 db.student.aggregate(  
2   [  
3     {$match : { } },  
4     {$group : { _id : "$name" , total : { $avg : "$marks" } } },  
5     {$sort : {total : 1} }  
6   ]  
7 )  
8
```

**Output:**

student 0.005 sec.	
_id	total
1 "Ketan"	60.0
2 "Amey"	76.5
3 "Gauri"	81.0
4 "Vijay"	92.5

### 3. min

Suppose we need minimum value from marks of each student in collection.

So we will set **\_id: "\$name"** which states that we will be grouping using **"name"** from each document.

And **total : { \$min : "\$marks" }** which state that we need minimum value of all "**marks**" which has same "**name**" value.

New Connection localhost:27017 itvoyagersDB

```
1 db.student.aggregate(  
2   [  
3     {$match : { } },  
4     {$group : { _id : "$name" , total : { $min : "$marks" } } }  
5   ]  
6 )  
7
```

Output:

	_id	total
1	Amey	73.0
2	Gauri	80.0
3	Ketan	60
4	Vijay	90.0

#### 4. max

Suppose we need maximum value from marks of each student in collection.

So we will set **\_id : "\$name"** which states that we will be grouping using "**name**" from each document.

And **total : { \$max : "\$marks" }** which state that we need maximum value of all "**marks**" which has same "**name**" value.

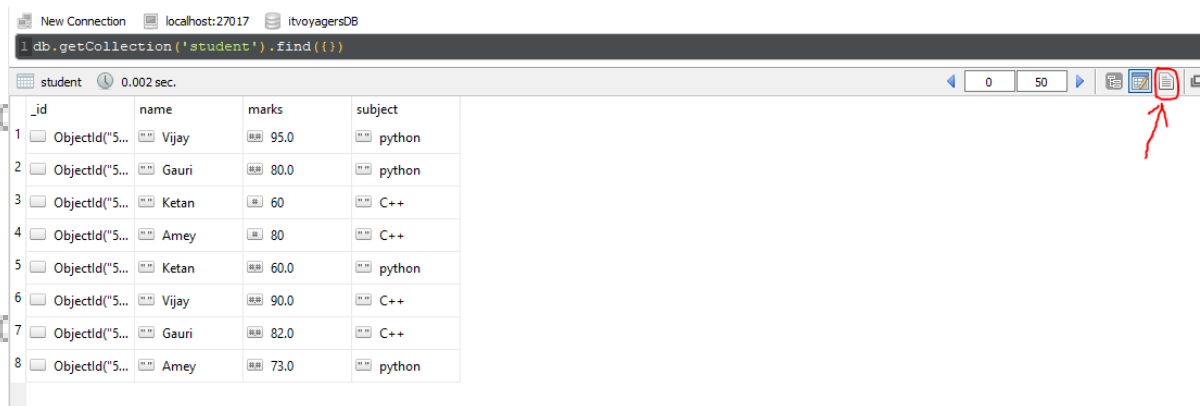
```
1 db.student.aggregate(  
2   [  
3     {$match : { } },  
4     {$group : { _id : "$name" , total : { $max : "$marks" } } }  
5   ]  
6 )  
7
```

Output:

	_id	total
1	Amey	80
2	Gauri	82.0
3	Ketan	60
4	Vijay	95

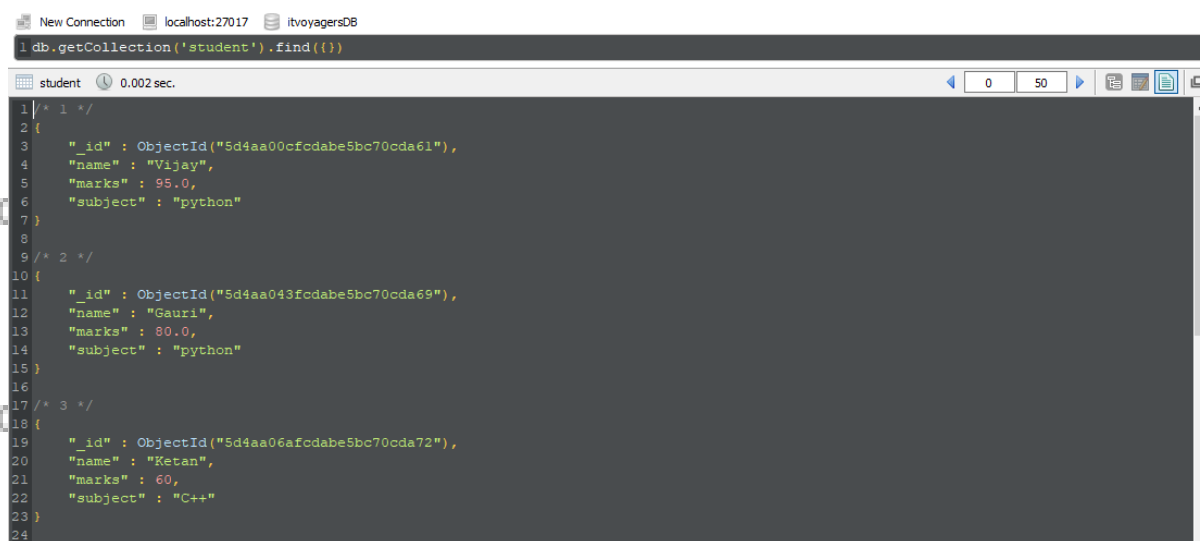
## Write a MongoDB query to use push and addToSet expression.

This is our collection in tabular format, let's view our collection in JSON format by clicking on **"View result on text mode"**.



	_id	name	marks	subject
1	ObjectId("5...")	Vijay	95.0	python
2	ObjectId("5...")	Gauri	80.0	python
3	ObjectId("5...")	Ketan	60	C++
4	ObjectId("5...")	Amey	80	C++
5	ObjectId("5...")	Ketan	60.0	python
6	ObjectId("5...")	Vijay	90.0	C++
7	ObjectId("5...")	Gauri	82.0	C++
8	ObjectId("5...")	Amey	73.0	python

We are going to perform push and addToSet operations on first document i.e. document which has **"name": "Vijay"**.



```

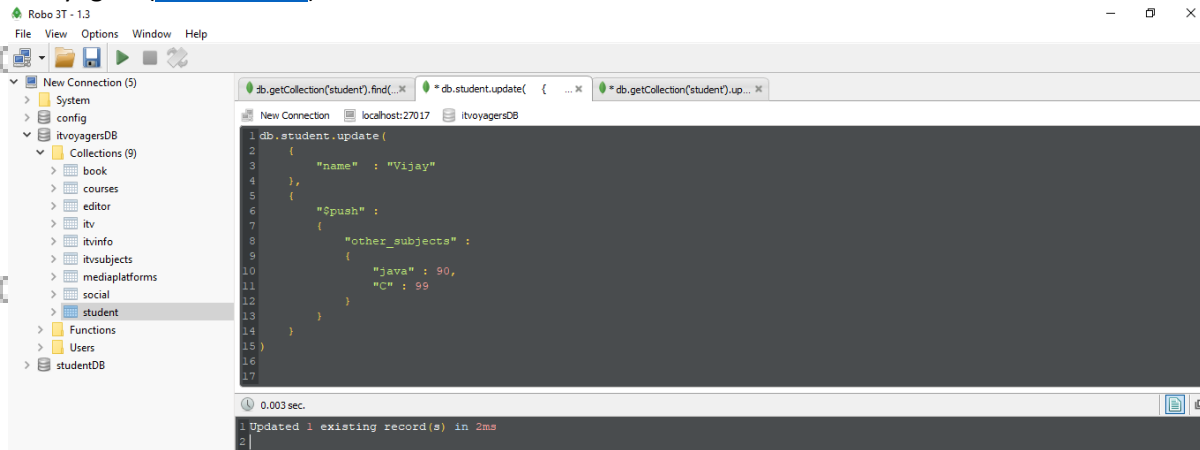
1 /* 1 */
2 {
3   "_id" : ObjectId("5d4aa00cfcdabe5bc70cda61"),
4   "name" : "Vijay",
5   "marks" : 95.0,
6   "subject" : "python"
7 }
8
9 /* 2 */
10 {
11   "_id" : ObjectId("5d4aa043fcdabe5bc70cda69"),
12   "name" : "Gauri",
13   "marks" : 80.0,
14   "subject" : "python"
15 }
16
17 /* 3 */
18 {
19   "_id" : ObjectId("5d4aa06afcdabe5bc70cda72"),
20   "name" : "Ketan",
21   "marks" : 60,
22   "subject" : "C++"
23 }
24

```

## \$push expression

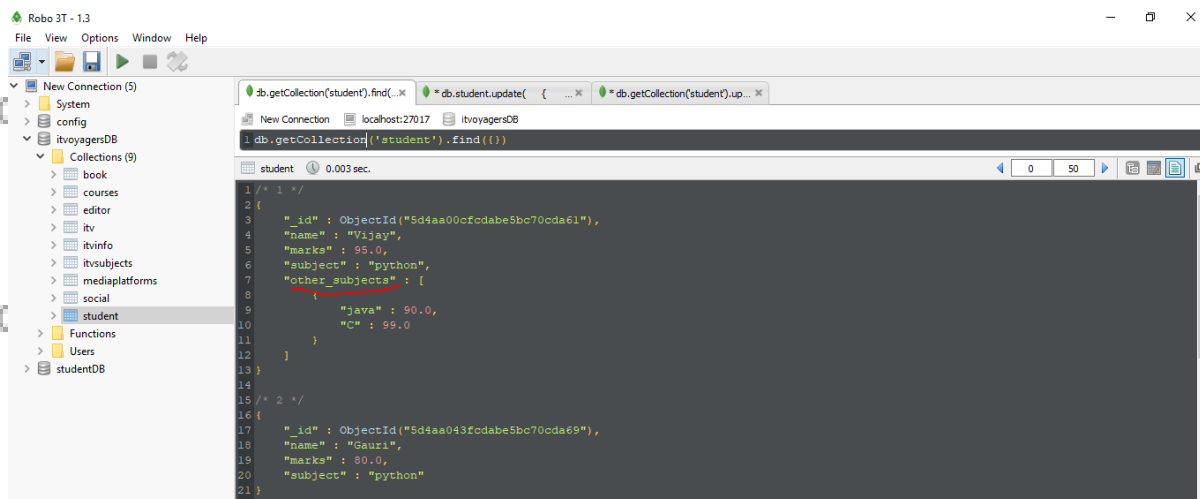
\$push is use with update method, it is use to add array element in the document.

In our example we are going to add **"other\_subjects"** array as element in our first document which has **"name": "Vijay"**. In first parameter of update method we will set **{"name": "Vijay"}** which is nothing but the condition on basis of which mongoDB will search document. We will add two key:values pair in **"other\_subjects"** array, execute the command.



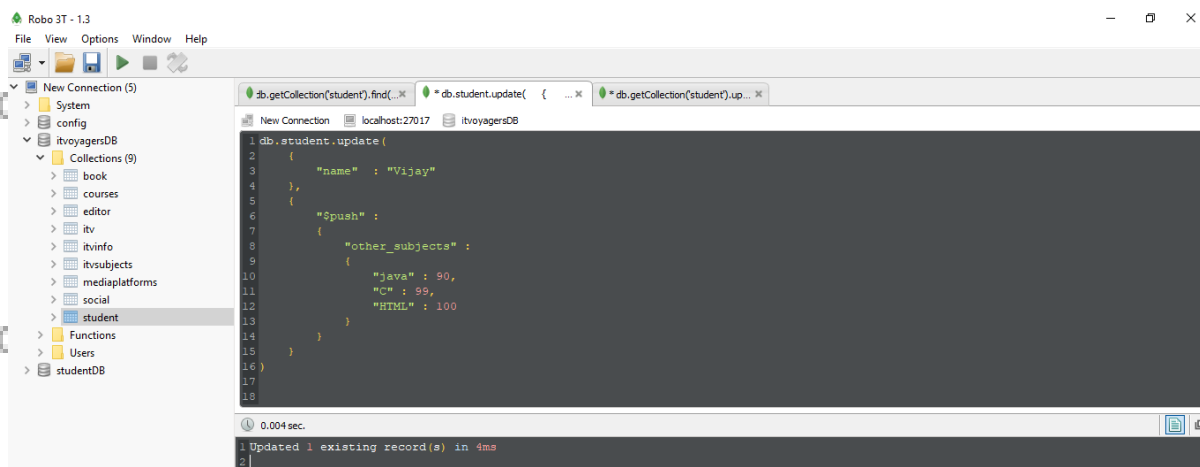
```
1 db.student.update(
2   {
3     "name" : "Vijay"
4   },
5   {
6     "$push" :
7     {
8       "other_subjects" :
9       {
10        "java" : 90,
11        "C" : 99
12      }
13    }
14  }
15 )
16
17
0.003 sec.
1 Updated 1 existing record(s) in 2ms
```

If we check all document from “**student**” collection, we will see that new array has been added in first document which has “**name**” : “**Vijay**”.



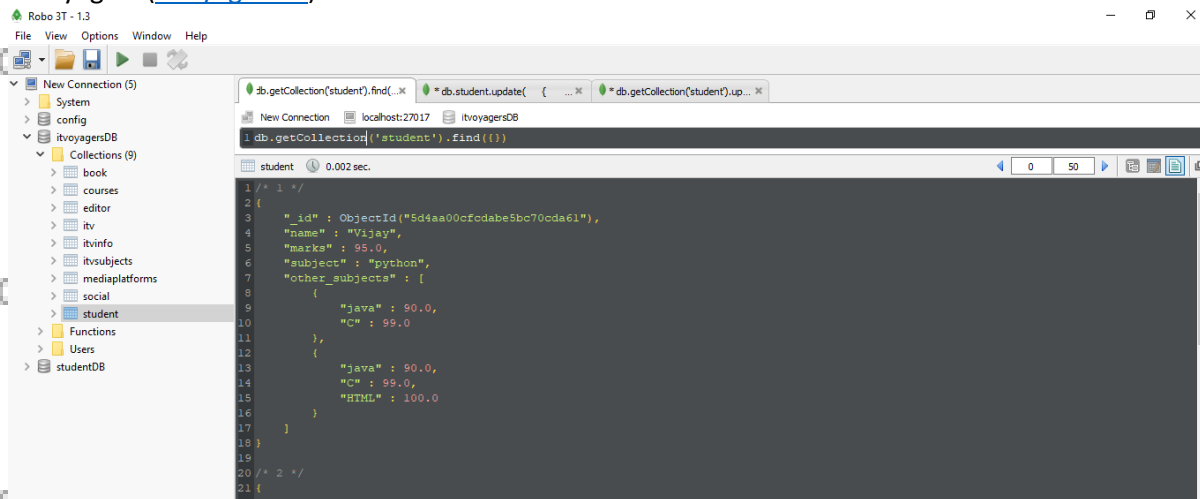
```
1 /* 1 */
2 {
3   "_id" : ObjectId("5d4aa00cfcdabe5bc70cda61"),
4   "name" : "Vijay",
5   "marks" : 95.0,
6   "subject" : "python",
7   "other_subjects" : {
8     "java" : 90.0,
9     "C" : 99.0
10  }
11 }
12
13
14
15 /* 2 */
16 {
17   "_id" : ObjectId("5d4aa043fcdabe5bc70cda69"),
18   "name" : "Gauri",
19   "marks" : 80.0,
20   "subject" : "python"
21 }
```

Let’s try to add one more key:value pair in same array, we have just added “**HTML**” : **100** in update command. Execute the command.



```
1 db.student.update(
2   {
3     "name" : "Vijay"
4   },
5   {
6     "$push" :
7     {
8       "other_subjects" :
9       {
10        "java" : 90,
11        "C" : 99,
12        "HTML" : 100
13      }
14    }
15  }
16 )
17
18
0.004 sec.
1 Updated 1 existing record(s) in 4ms
```

And now if we view the document we will see that instead of adding “**HTML**” : **100** as element in “**other\_subjects**” array MongoDB has added all three key:value pairs as another element in “**other\_subjects**” array.



This is how \$push works it will add array element in document and if array element is already present then it will append all the values as separate element in it.

## \$addToSet expression

\$addToSet will add array element in document, but unlike \$push if the array is already present in document then it will update in array and add the value in array.

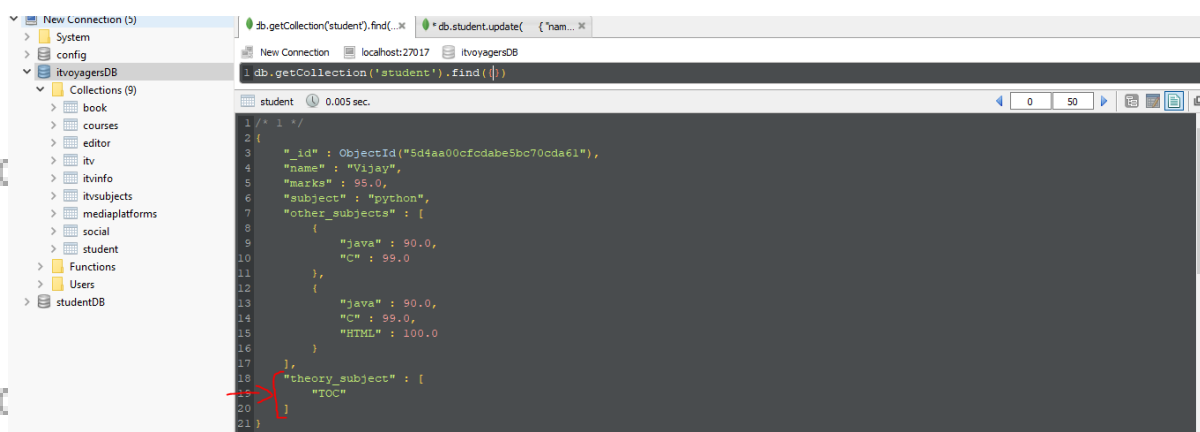
In our example we are going to add **"theory\_subject"** array as element in our first document which has **"name" : "Vijay"**. In first parameter of update method we will set **{"name" : "Vijay"}** which is nothing but the condition on basis of which mongoDB will search document. We will add one value in **"theory\_subject"** array, execute the command.

```

1 db.student.update(
2   { "name" : "Vijay" },
3   {
4     $addToSet : { "theory_subject" : "TOC" }
5   }
6 )
7

```

## Output:



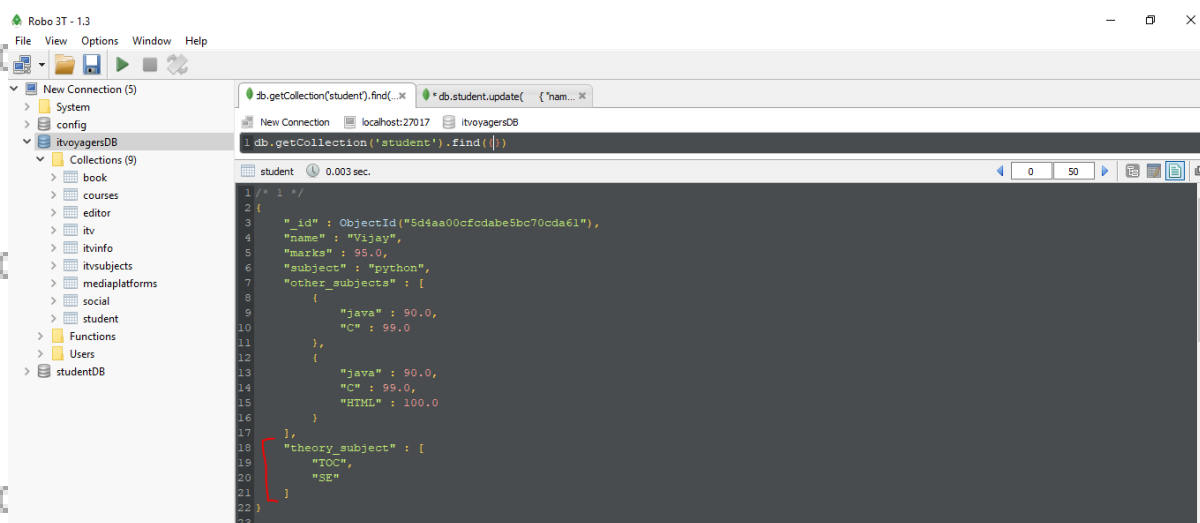
If we check all document from “**student**” collection, we will see that new array has been added in first document which has “**name**”: “**Vijay**”.

Let’s try adding one more value in same array, we have just change “**TOC**” to “**SE**” in update command. Execute the command.

```
New Connection localhost:27017 itvoyagersDB

1 db.student.update (
2   { "name" : "Vijay" },
3   {
4     $addToSet : { "theory_subject" : "SE" }
5   }
6 )
7
```

Output:



```
Robo 3T - 1.3
File View Options Window Help

New Connection (5)
  System
  config
  itvoyagersDB
    Collections (9)
      book
      courses
      editor
      itv
      itvinfo
      itvsubjects
      mediaplatforms
      social
      student
      Functions
      Users
      studentDB

db.getCollection('student').find(...)
db.student.update( { 'name' : 'Vijay' }, { $addToSet : { 'theory_subject' : 'SE' } })

student 0.003 sec.

1 / 1
2 {
3   "_id" : ObjectId("5d4aa00cfcdabe5bc70cda61"),
4   "name" : "Vijay",
5   "marks" : 95.0,
6   "subject" : "python",
7   "other_subjects" : [
8     {
9       "java" : 90.0,
10      "C" : 99.0
11    },
12    {
13      "java" : 90.0,
14      "C" : 99.0,
15      "HIML" : 100.0
16    }
17  ],
18   "theory_subject" : [
19     "TOC",
20     "SE"
21   ]
22 }
23
```

We can see that now in “**theory\_subject**” array another element has been added, unlike \$push which adds all the value as separate element.

\$addToSet will add array element in document and if it is present then it will update it, Now let’s add one more array “**math\_subject**” in same document.

```
New Connection localhost:27017 itvoyagersDB

1 db.student.update (
2   { "name" : "Vijay" },
3   {
4     $addToSet : { "math_subject" : "DM" }
5   }
6 )
7
```

## Output:

Robo 3T - 1.3

File View Options Window Help

New Connection (5)

- System
- config
- itvoyagersDB
  - Collections (9)
    - book
    - courses
    - editor
    - itr
    - itrinfo
    - itrsubjects
    - mediaplatforms
    - social
    - student
  - Functions
  - Users
  - studentDB

db.getCollection('student').find({})

New Connection localhost:27017 itvoyagersDB

1 db.getCollection('student').find({})

student 0.002 sec.

```

1 /* 1 */
2 {
3   "_id" : ObjectId("5d4aa00cf0dabe5bc70cda61"),
4   "name" : "Vijay",
5   "marks" : 95.0,
6   "subject" : "python",
7   "other_subjects" : [
8     {
9       "java" : 90.0,
10      "C" : 99.0
11    },
12    {
13      "java" : 90.0,
14      "C" : 99.0,
15      "HTML" : 100.0
16    }
17  ],
18   "theory_subject" : [
19     "TOC",
20     "SSE"
21   ],
22   "math_subject" : [
23     "DM"
24   ]
25 }
26

```

**Write a MongoDB query to use first and last expression.**

**Just view our student collection documents**

New Connection localhost:27017 itvoyagersDB

1 db.getCollection('student').find({})

student 0.004 sec.

	_id	name	marks	subject	other_subjects	theory_subject	math_subject
1	ObjectId("5...")	Vijay	95.0	python	[ 2 elements ]	[ 2 elements ]	[ 1 element ]
2	ObjectId("5...")	Gauri	80.0	python			
3	ObjectId("5...")	Ketan	60	C++			
4	ObjectId("5...")	Amey	80	C++			
5	ObjectId("5...")	Ketan	60.0	python			
6	ObjectId("5...")	Vijay	90.0	C++			
7	ObjectId("5...")	Gauri	82.0	C++			
8	ObjectId("5...")	Amey	73.0	python			

**This is our collection**

student 0.004 sec.

	_id	name	marks	subject	other_subjects	theory_subject	math_subject
1	ObjectId("5...")	Vijay	95.0	python	[ 2 elements ]	[ 2 elements ]	[ 1 element ]
2	ObjectId("5...")	Gauri	80.0	python			
3	ObjectId("5...")	Ketan	60	C++			
4	ObjectId("5...")	Amey	80	C++			
5	ObjectId("5...")	Ketan	60.0	python			
6	ObjectId("5...")	Vijay	90.0	C++			
7	ObjectId("5...")	Gauri	82.0	C++			
8	ObjectId("5...")	Amey	73.0	python			

## \$first expression

Now we know that each student has 2 document one for subject python and another one for C++. We want first document's marks of all the students. We want those highlighted which are first entry with respect to "**name**" in all documents.

student 0.047 sec.

_id	name	marks	subject	other_subjects	theory_subject	math_subject
1	Vijay	95.0	python	[ 2 elements ]	[ 2 elements ]	[ 1 element ]
2	Gauri	80.0	python			
3	Ketan	60	C++			
4	Amey	80	C++			
5	Ketan	60.0	python			
6	Vijay	90.0	C++			
7	Gauri	82.0	C++			
8	Amey	73.0	python			

We can use \$first to get those marks.

Now in below operations we will use \$group and \$first expression.

**\$group** : In \$group we will set "**\_id**" to "**\$name**" which states that we will be performing operation on "**name**" key of all documents.

**\$first** : In we will pass "**\$marks**" which states that it will retrieve first document mark from documents which has same "**name**" value.

And first\_mark is nothing but the column name in which result is going to be displayed.

New Connection localhost:27017 itvoyagersDB

```

1 db.student.aggregate (
2   [
3     {
4       $group : {
5         _id : "$name" ,
6         first_entry : {$first : "$marks"}
7       }
8     }
9   ]
10 )

```

**Output:**

student 0.004 sec.	
_id	first_entry
1 Amey	80
2 Gauri	80.0
3 Ketan	60
4 Vijay	95.0

**\$last expression**

As the name suggest it will retrieve last document with respect to search condition. In our case it will retrieve last document mark which has same "name" value. Those are the last marks with respect to "name".

student 0.047 sec.						
_id	name	marks	subject	other_subjects	theory_subject	math_subject
1 ObjectId("5...")	Vijay	95.0	python	[ 2 elements ]	[ 2 elements ]	[ 1 element ]
2 ObjectId("5...")	Gauri	80.0	python			
3 ObjectId("5...")	Ketan	60	C++			
4 ObjectId("5...")	Amey	80	C++			
5 ObjectId("5...")	Ketan	60.0	python			
6 ObjectId("5...")	Vijay	90.0	C++			
7 ObjectId("5...")	Gauri	82.0	C++			
8 ObjectId("5...")	Amey	73.0	python			

There is only one change in query i.e. instead of \$first we will use \$last.

New Connection localhost:27017 itvoyagersDB

```

1 db.student.aggregate (
2   [
3     {
4       $group : {
5         _id : "$name" ,
6         first_entry : {$last : "$marks"}
7       }
8     }
9   ]
10 )

```

**Output:**

student 0.002 sec.	
_id	first_entry
1 Amey	73.0
2 Gauri	82.0
3 Ketan	60.0
4 Vijay	90.0

Let's fetch first marks with respect to subject, so that we will be getting those marks in return, because those are the first entry with respect to subject i.e. python and c++.

New Connection

localhost:27017

itvoyagersDB

```
1 db.getCollection('student').find({})
```

student

0.047 sec.

	_id	name	marks	subject	other_subjects	theory_subject	math_subject
1	ObjectId("5...")	Vijay	95.0	python	[ 2 elements ]	[ 2 elements ]	[ 1 element ]
2	ObjectId("5...")	Gauri	80.0	python			
3	ObjectId("5...")	Ketan	60	C++			
4	ObjectId("5...")	Amey	80	C++			
5	ObjectId("5...")	Ketan	60.0	python			
6	ObjectId("5...")	Vijay	90.0	C++			
7	ObjectId("5...")	Gauri	82.0	C++			
8	ObjectId("5...")	Amey	73.0	python			

We will set **\_id: "\$subject"** and we will use **\$first** to get our result.

New Connection localhost:27017 itvoyagersDB

```
1 db.student.aggregate (
2   [
3     {
4       $group : {
5         _id : "$subject" ,
6         first_entry : {$first : "$marks"}
7       }
8     }
9   ]
10 )
11
```

**Output:**

student 0.002 sec.	
_id	first_entry
1 C++	60
2 python	95.0