

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»  
Факультет инфокоммуникационных технологий

**ОТЧЕТ**  
**О ЛАБОРАТОРНОЙ РАБОТЕ № 6**  
по теме:

*Работа с БД в СУБД MongoDB*

по дисциплине: Проектирование и реализация баз данных

Специальность:  
09.03.03 Мобильные и сетевые технологии

Проверила:  
Говорова М.М.  
Дата: \_\_\_\_\_ 2023 г.  
Оценка \_\_\_\_\_

Выполнила:  
студентка группы К3239  
Борисова Э. Е

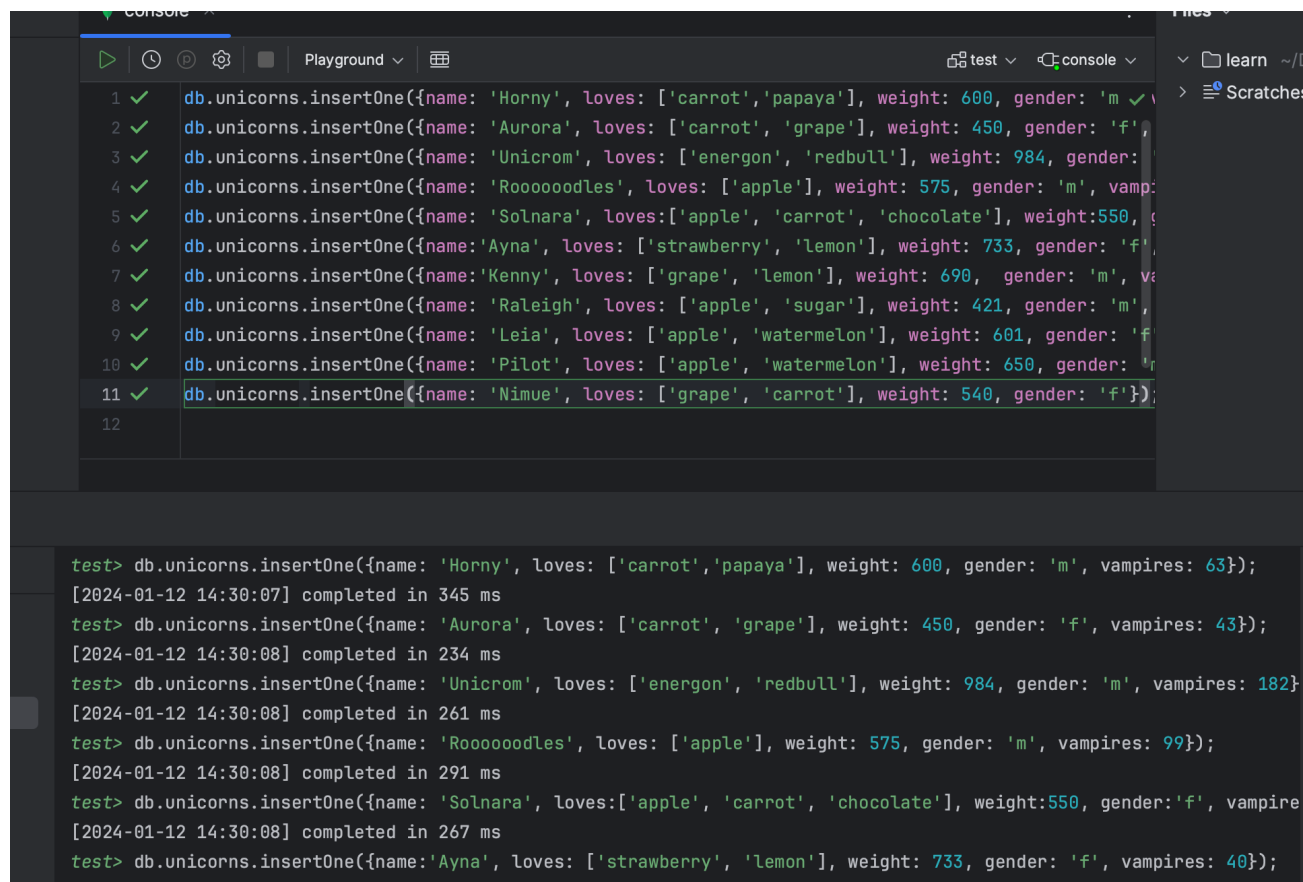
**Цель:** овладеть практическими навыками работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД MongoDB 4+, 6.0.6 (текущая).

### Практическое задание 2.1.1:

1. Создайте базу данных *learn*.
2. Заполните коллекцию единорогов *unicorns*:



The screenshot displays a MongoDB Playground interface. The top section shows a list of 11 MongoDB insertOne commands for the 'unicorns' collection, each with a green checkmark indicating successful execution. The commands insert data for unicorns named Horny, Aurora, Unicrom, Roooooodles, Solnara, Ayna, Kenny, Raleigh, Leia, Pilot, and Nimue, each with specific loves, weights, genders, and vampire counts. The bottom section shows the corresponding test commands and their execution times, confirming the successful insertion of each document.

```
1 db.unicorns.insertOne({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
2 db.unicorns.insertOne({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
3 db.unicorns.insertOne({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
4 db.unicorns.insertOne({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
5 db.unicorns.insertOne({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampire: 40});
6 db.unicorns.insertOne({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
7 db.unicorns.insertOne({name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 40});
8 db.unicorns.insertOne({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 40});
9 db.unicorns.insertOne({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 40});
10 db.unicorns.insertOne({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'f', vampires: 40});
11 db.unicorns.insertOne({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f', vampires: 40});

test> db.unicorns.insertOne({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
[2024-01-12 14:30:07] completed in 345 ms
test> db.unicorns.insertOne({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
[2024-01-12 14:30:08] completed in 234 ms
test> db.unicorns.insertOne({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
[2024-01-12 14:30:08] completed in 261 ms
test> db.unicorns.insertOne({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
[2024-01-12 14:30:08] completed in 291 ms
test> db.unicorns.insertOne({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampire: 40});
[2024-01-12 14:30:08] completed in 267 ms
test> db.unicorns.insertOne({name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});
[2024-01-12 14:30:08] completed in 267 ms
```

3. Используя второй способ, вставьте в коллекцию единорогов документ:

The screenshot shows a MongoDB Playground interface. The code editor contains the following code:

```
1 document= ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampir ✓ :  
2 db.users.insertOne(document)  
3  
4
```

The output panel shows the result of the insertion as a table with 5 columns: gender, loves, name, vampires, and weight. The table contains one row of data:

	gender	loves	name	vampires	weight
1	m	["grape", "watermelon"]	Dunx	165	704

4. Проверьте содержимое коллекции с помощью метода find.

The screenshot shows a MongoDB Playground interface. The code editor contains the following code:

```
1 db.unicorns.find()
```

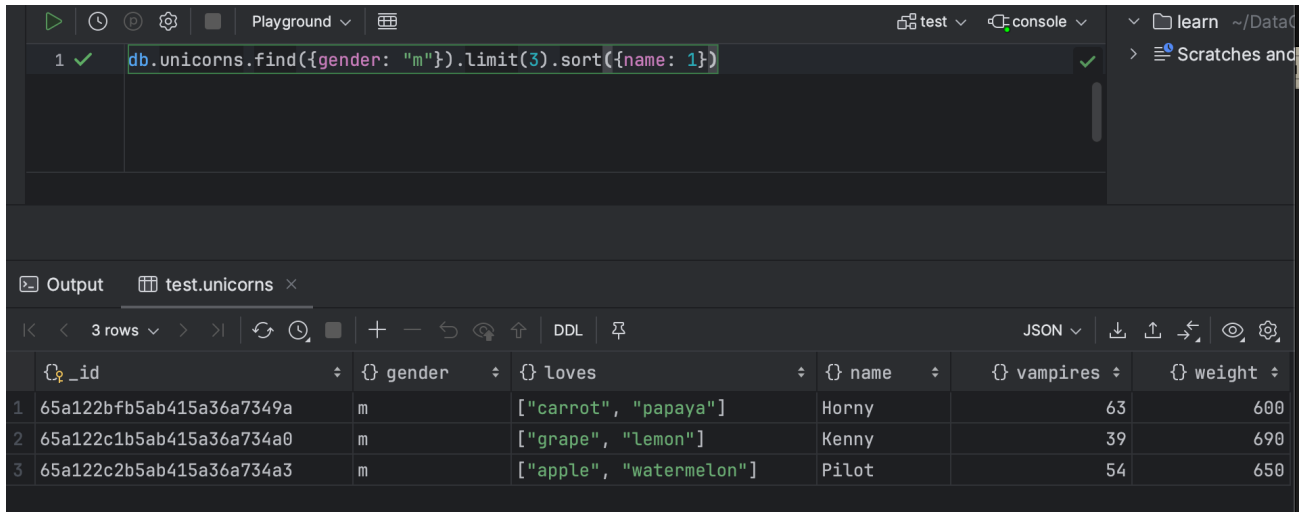
The output panel shows the result of the find query as a table with 6 columns: \_id, gender, loves, name, and vampires. The table contains 11 rows of data:

	_id	gender	loves	name	vampires
1	65a122bfb5ab415a36a7349a	m	["carrot", "papaya"]	Horny	6
2	65a122bfb5ab415a36a7349b	f	["carrot", "grape"]	Aurora	4
3	65a122c0b5ab415a36a7349c	m	["energon", "redbull"]	Unicrom	18
4	65a122c0b5ab415a36a7349d	m	["apple"]	Rooodoodles	9
5	65a122c0b5ab415a36a7349e	f	["apple", "carrot", "chocolate"]	Solnara	8
6	65a122c1b5ab415a36a7349f	f	["strawberry", "lemon"]	Ayna	4
7	65a122c1b5ab415a36a734a0	m	["grape", "lemon"]	Kenny	3
8	65a122c1b5ab415a36a734a1	m	["apple", "energon"]	Releish	
9	65a122c1b5ab415a36a734a2	f	["apple", "energon"]	Releish	
10	65a122c1b5ab415a36a734a3	f	["apple", "energon"]	Releish	
11	65a122c1b5ab415a36a734a4	f	["apple", "energon"]	Releish	

## Практическое задание 2.2.1:

1. Сформируйте запросы для вывода списков самцов и самок единорогов. Ограничьте список самок первыми тремя особями. Отсортируйте списки по имени.

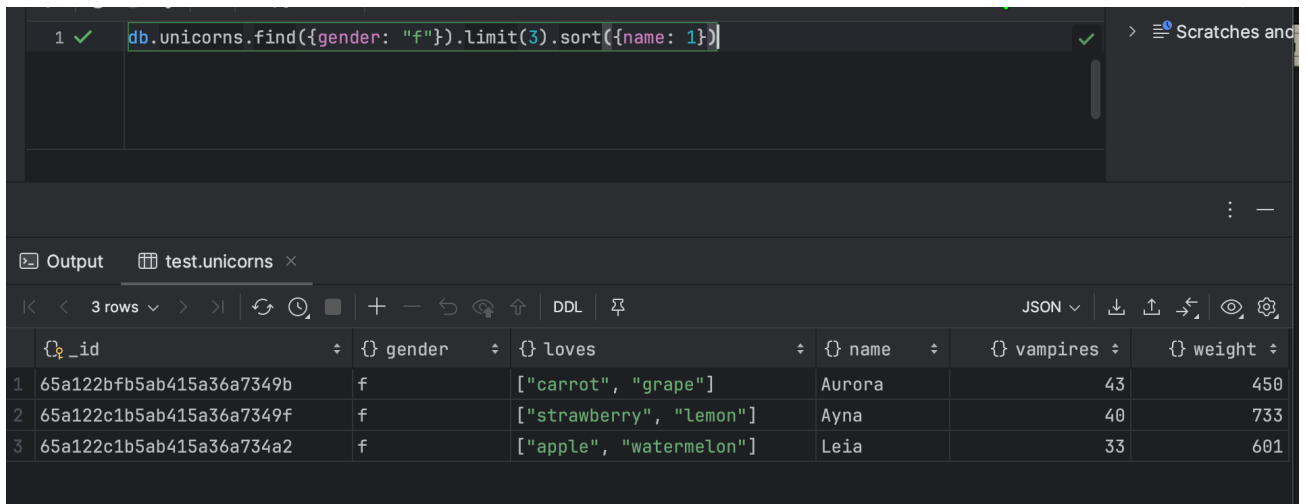
Для самцов:



The screenshot shows a MongoDB Playground interface. The query entered is `db.unicorns.find({gender: "m"}).limit(3).sort({name: 1})`. The results are displayed in a table with 7 columns: `_id`, `gender`, `loves`, `name`, `vampires`, and `weight`.

	<code>_id</code>	<code>gender</code>	<code>loves</code>	<code>name</code>	<code>vampires</code>	<code>weight</code>
1	65a122bfb5ab415a36a7349a	m	["carrot", "papaya"]	Horny	63	600
2	65a122c1b5ab415a36a734a0	m	["grape", "lemon"]	Kenny	39	690
3	65a122c2b5ab415a36a734a3	m	["apple", "watermelon"]	Pilot	54	650

Самки:



The screenshot shows a MongoDB Playground interface. The query entered is `db.unicorns.find({gender: "f"}).limit(3).sort({name: 1})`. The results are displayed in a table with 7 columns: `_id`, `gender`, `loves`, `name`, `vampires`, and `weight`.

	<code>_id</code>	<code>gender</code>	<code>loves</code>	<code>name</code>	<code>vampires</code>	<code>weight</code>
1	65a122bfb5ab415a36a7349b	f	["carrot", "grape"]	Aurora	43	450
2	65a122c1b5ab415a36a7349f	f	["strawberry", "lemon"]	Ayna	40	733
3	65a122c1b5ab415a36a734a2	f	["apple", "watermelon"]	Leia	33	601

2. Найдите всех самок, которые любят carrot. Ограничьте этот список первой особью с помощью функций `findOne` и `limit`.

The screenshot shows a MongoDB Playground interface. The code editor contains the query `db.unicorns.find({gender: 'f'}, {_id: 0, name: 1, loves: 'carrot'})`. The output window, titled "test.unicorns", displays 5 rows of results. The table has columns for index, loves, and name.

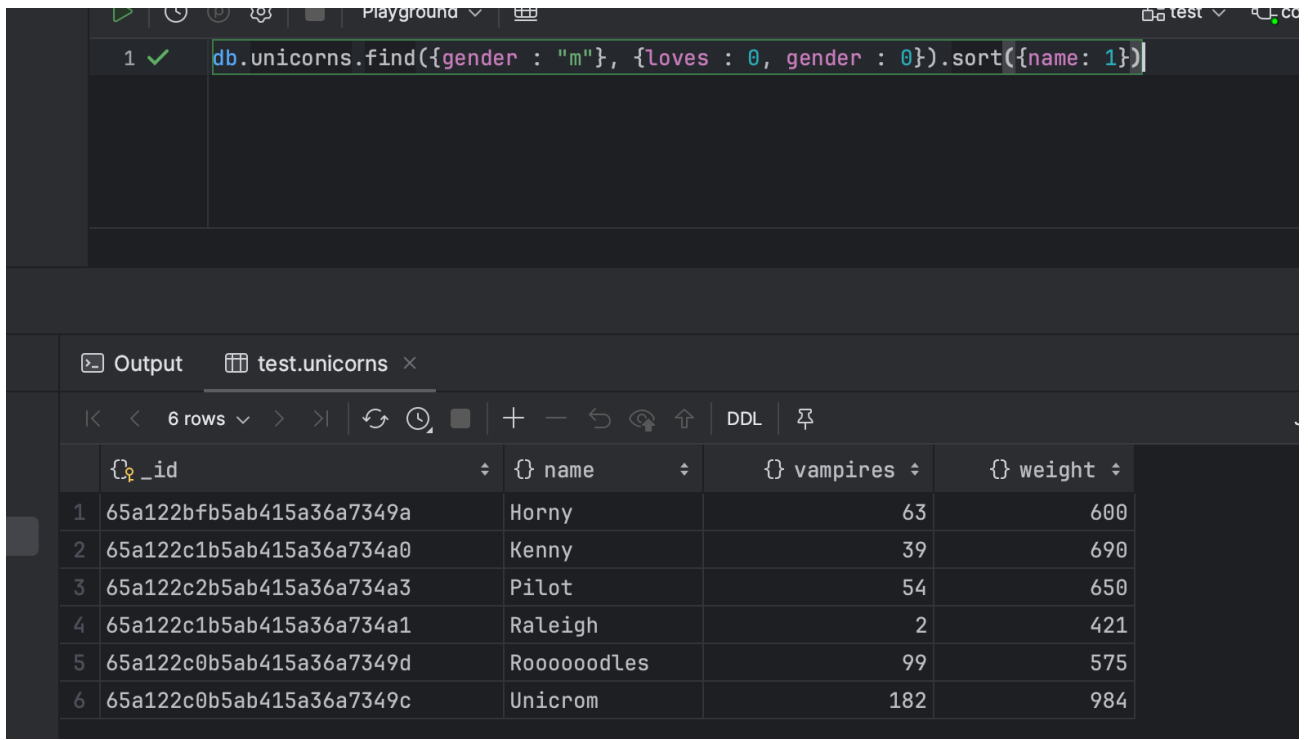
	loves	name
1	carrot	Aurora
2	carrot	Solnara
3	carrot	Ayna
4	carrot	Leia
5	carrot	Nimue

The screenshot shows the same MongoDB Playground interface, but the code editor now includes the `limit(1)` method: `db.unicorns.find({gender: 'f'}, {_id: 0, name: 1, loves: 'carrot'}).limit(1)`. The output window, titled "test.unicorns", now displays only 1 row of results.

	loves	name
1	carrot	Aurora

### Практическое задание 2.2.2:

Модифицируйте запрос для вывода списков самцов единорогов, исключив из результата информацию о предпочтениях и поле.

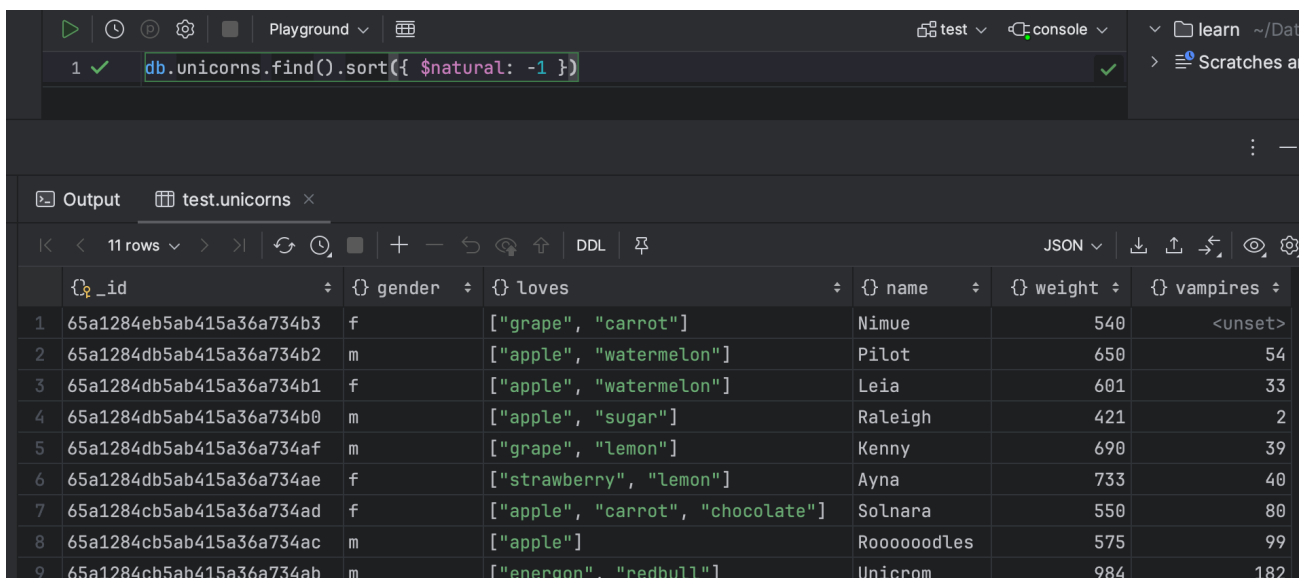


The screenshot shows a MongoDB Playground interface. The query entered is `db.unicorns.find({gender: "m"}, {loves: 0, gender: 0}).sort({name: 1})`. The results are displayed in a table with 6 rows, sorted by name in ascending order.

	_id	name	vampires	weight
1	65a122bfb5ab415a36a7349a	Horny	63	600
2	65a122c1b5ab415a36a734a0	Kenny	39	690
3	65a122c2b5ab415a36a734a3	Pilot	54	650
4	65a122c1b5ab415a36a734a1	Raleigh	2	421
5	65a122c0b5ab415a36a7349d	Roooooodles	99	575
6	65a122c0b5ab415a36a7349c	Unicrom	182	984

### Практическое задание 2.2.3:

Вывести список единорогов в обратном порядке добавления.



The screenshot shows a MongoDB Playground interface. The query entered is `db.unicorns.find().sort({ $natural: -1 })`. The results are displayed in a table with 11 rows, sorted by natural order in descending order.

	_id	gender	loves	name	weight	vampires
1	65a1284eb5ab415a36a734b3	f	["grape", "carrot"]	Nimue	540	<unset>
2	65a1284db5ab415a36a734b2	m	["apple", "watermelon"]	Pilot	650	54
3	65a1284db5ab415a36a734b1	f	["apple", "watermelon"]	Leia	601	33
4	65a1284db5ab415a36a734b0	m	["apple", "sugar"]	Raleigh	421	2
5	65a1284db5ab415a36a734af	m	["grape", "lemon"]	Kenny	690	39
6	65a1284db5ab415a36a734ae	f	["strawberry", "lemon"]	Ayna	733	40
7	65a1284cb5ab415a36a734ad	f	["apple", "carrot", "chocolate"]	Solnara	550	80
8	65a1284cb5ab415a36a734ac	m	["apple"]	Roooooodles	575	99
9	65a1284cb5ab415a36a734ab	m	["energon", "redbull"]	Unicrom	984	182

### Практическое задание 2.1.4:

Вывести список единорогов с названием первого любимого предпочтения, исключив идентификатор.

The screenshot shows a MongoDB Playground interface. The code editor at the top contains the query: `db.unicorns.find({}, {loves : {$slice : 1}, _id : 0})`. Below the editor, the 'Output' tab is selected, displaying a table with 11 rows. The table has columns for gender, loves, name, vampires, and weight.

	gender	loves	name	vampires	weight
1	m	["carrot"]	Horny	63	600
2	f	["carrot"]	Aurora	43	450
3	m	["energon"]	Unicrom	182	984
4	m	["apple"]	Rooodoodles	99	575
5	f	["apple"]	Solnara	80	550
6	f	["strawberry"]	Ayna	40	733
7	m	["grape"]	Kenny	39	690
8	m	["apple"]	Raleigh	2	421
9	f	["apple"]	Leia	33	601
10	m	["apple"]	Pilot	54	650

### Практическое задание 2.3.1:

Вывести список самок единорогов весом от полутонны до 700 кг, исключив вывод идентификатора.

The screenshot shows a MongoDB Playground interface. The code editor at the top contains the query: `db.unicorns.find({gender : "f", weight : {$gte : 500, $lte : 700}}, {_id: 0})`. Below the editor, the 'Output' tab is selected, displaying a table with 3 rows. The table has columns for gender, loves, name, vampires, and weight.

	gender	loves	name	vampires	weight
1	f	["apple", "carrot", "chocolate"]	Solnara	80	550
2	f	["apple", "watermelon"]	Leia	33	601
3	f	["grape", "carrot"]	Nimue	<unset>	540

### Практическое задание 2.3.3:

Найти всех единорогов, не имеющих ключ *vampires*.

The screenshot shows a MongoDB Playground interface. At the top, the query `db.unicorns.find({'vampires': {'$exists': false}})` is entered and executed successfully. Below the query, the 'Output' tab is selected, displaying a table with the results of the query. The table has columns for `_id`, `gender`, `loves`, `name`, and `weight`. One document is shown with `_id` 65a1284eb5ab415a36a734b3, `gender` f, `loves` ["grape", "carrot"], `name` Nimue, and `weight` 540.

	_id	gender	loves	name	weight
1	65a1284eb5ab415a36a734b3	f	["grape", "carrot"]	Nimue	540

### Практическое задание 2.3.4:

Вывести список упорядоченный список имен самцов единорогов с информацией об их первом предпочтении.

The screenshot shows a MongoDB Playground interface. At the top, the query `db.unicorns.find({'gender': 'm'}, {'loves': {'$slice': 1}})` is entered and executed successfully. Below the query, the 'Output' tab is selected, displaying a table with the results of the query. The table has columns for `_id`, `gender`, `loves`, `name`, `vampires`, and `weight`. Six documents are shown, all with `gender` m, sorted by `loves`.

	_id	gender	loves	name	vampires	weight
1	65a1284bb5ab415a36a734a9	m	["carrot"]	Horny	63	600
2	65a1284cb5ab415a36a734ab	m	["energon"]	Unicrom	182	984
3	65a1284cb5ab415a36a734ac	m	["apple"]	Roooooodles	99	575
4	65a1284db5ab415a36a734af	m	["grape"]	Kenny	39	690
5	65a1284db5ab415a36a734b0	m	["apple"]	RaLeigh	2	421
6	65a1284db5ab415a36a734b2	m	["apple"]	Pilot	54	650

### Практическое задание 3.1.1:

Создайте коллекцию *towns*, включающую следующие документы:

```
db.towns.insertMany([
  {
    name: "Punxsutawney ",
    populatiuon: 6200,
    last_sensus: ISODate("2008-01-31"),
    famous_for: [],
    mayor: {
      name: "Jim Wehrle"
    }
  },
  {
    name: "New York",
    populatiuon: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["status of liberty", "food"],
    mayor: {
      name: "Michael Bloomberg",
      party: "I"
    }
  }
])
```



```
{name: "Portland",
populatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
name: "Sam Adams",
party: "D"}}
]);
```

1. Сформировать запрос, который возвращает список городов с независимыми мэрами (party="I"). Вывести только название города и информацию о мэре.

The screenshot shows the MongoDB Playground interface. The query entered is `db.towns.find({"mayor.party": "I"}, {name: 1, mayor: 1})`. The result is displayed in a table with columns `_id`, `mayor`, and `name`. Only one row is shown, corresponding to New York.

	_id	mayor	name
1	65a171f0c7dd9e6803b776ac	{"name": "Michael Bloomberg", "party": "I"}	New York

2. Сформировать запрос, который возвращает список беспартийных мэров (party отсутствует). Вывести только название города и информацию о мэре.

The screenshot shows the MongoDB Playground interface. The query entered is `db.towns.find({"mayor.party": {"$exists": 1}}, {name: 1, mayor: 1, _id: false})`. The result is displayed in a table with columns `mayor` and `name`. Two rows are shown, corresponding to New York and Portland.

	mayor	name
1	{"name": "Michael Bloomberg", "party": "I"}	New York
2	{"name": "Sam Adams", "party": "D"}	Portland

### **Практическое задание 3.1.2:**

1. Сформировать функцию для вывода списка самцов единорогов.

```

1 ✓ function maleUnicorns_list() {return db.unicorns.find({gender:'m'})}
2 ✓ maleUnicorns_list()

```

Output Result 38 Result 38-2 ×

6 rows

	{ _id	{ gender	{ loves	{ name	{ vampires	{ weight
1	65a1284bb5ab415a36a734a9	m	["carrot", "papaya"]	Horny	63	600
2	65a1284cb5ab415a36a734ab	m	["energon", "redbull"]	Unicrom	182	984
3	65a1284cb5ab415a36a734ac	m	["apple"]	Rooodooodles	99	575
4	65a1284db5ab415a36a734af	m	["grape", "lemon"]	Kenny	39	690
5	65a1284db5ab415a36a734b0	m	["apple", "sugar"]	Raleigh	2	421
6	65a1284db5ab415a36a734b2	m	["apple", "watermelon"]	Pilot	54	650

2. Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.

3. Вывести результат, используя `forEach`.

```

1 ✓ var cursor = maleUnicorns_list().sort({name:1}).limit(2);
2
3 ✓ cursor.forEach(function(obj){print(obj.name)})

```

Output Result 43-2 × Result 42-2

2 rows

	{ 0
1	Dunx
2	Horny

### Практическое задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

The screenshot shows the MongoDB Playground interface. The query editor contains the following code:

```
1 db.unicorns.find({gender : "f", weight : {$gte : 500, $lte : 600}}).count()
```

The results panel shows the output of the query:

	result
1	2

### Практическое задание 3.2.2:

Вывести список предпочтений.

The screenshot shows the MongoDB Playground interface. The query editor contains the following code:

```
1 db.unicorns.distinct("loves")
```

The results panel shows the output of the query:

	result
1	apple
2	carrot
3	chocolate
4	energon
5	grape
6	lemon
7	papaya
8	redbull
9	strawberry
10	sugar
11	watermelon

### Практическое задание 3.2.3:

Посчитать количество особей единорогов обоих полов.

The screenshot shows the MongoDB Playground interface. The query editor contains the following code:

```
1 db.unicorns.aggregate({"$group":{"_id":"$gender", count : {$sum: 1}}})
```

The results panel shows the output of the query:

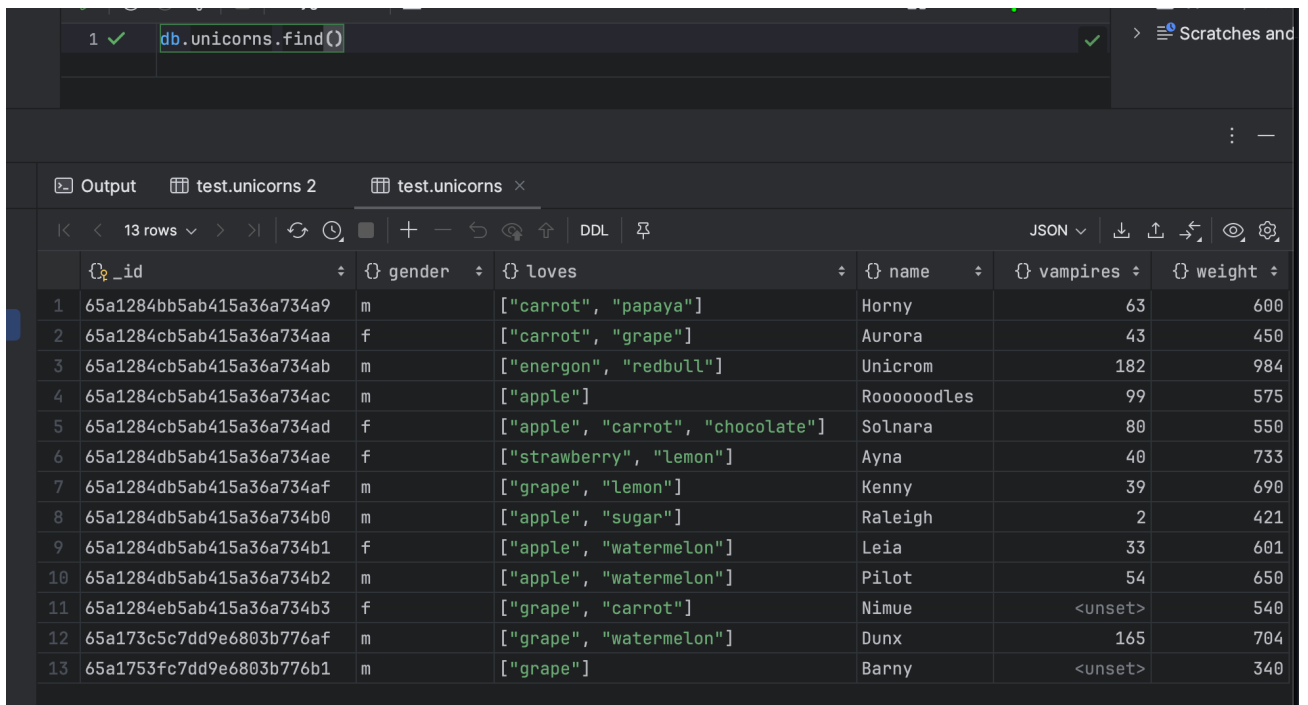
	_id	count
1	m	7
2	f	5

### Практическое задание 3.3.1:

1. Выполнить команду:

```
> db.unicorns.save({name: 'Barney', loves: ['grape'],  
weight: 340, gender: 'm'})
```

2. Проверить содержимое коллекции *unicorns*.



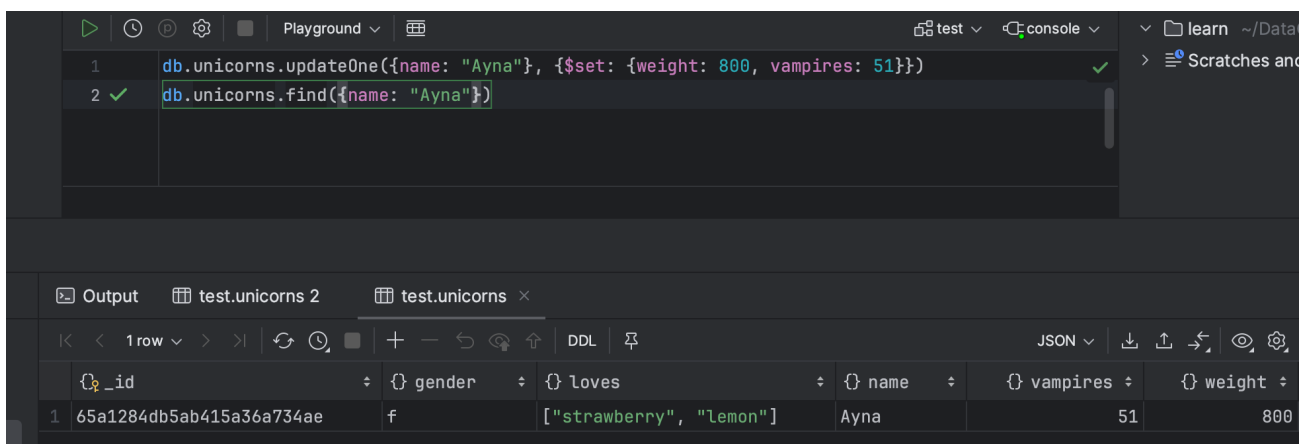
The screenshot shows the MongoDB Playground interface. The command `db.unicorns.find()` has been executed successfully. Below the command, a table displays the contents of the `unicorns` collection. The table has 13 rows, each representing a unicorn with fields: `_id`, `gender`, `loves`, `name`, `vampires`, and `weight`.

	<code>_id</code>	<code>gender</code>	<code>loves</code>	<code>name</code>	<code>vampires</code>	<code>weight</code>
1	65a1284bb5ab415a36a734a9	m	["carrot", "papaya"]	Horny	63	600
2	65a1284cb5ab415a36a734aa	f	["carrot", "grape"]	Aurora	43	450
3	65a1284cb5ab415a36a734ab	m	["energon", "redbull"]	Unicrom	182	984
4	65a1284cb5ab415a36a734ac	m	["apple"]	Rooooooodles	99	575
5	65a1284cb5ab415a36a734ad	f	["apple", "carrot", "chocolate"]	Solnara	80	550
6	65a1284db5ab415a36a734ae	f	["strawberry", "lemon"]	Ayna	40	733
7	65a1284db5ab415a36a734af	m	["grape", "lemon"]	Kenny	39	690
8	65a1284db5ab415a36a734b0	m	["apple", "sugar"]	Raleigh	2	421
9	65a1284db5ab415a36a734b1	f	["apple", "watermelon"]	Leia	33	601
10	65a1284db5ab415a36a734b2	m	["apple", "watermelon"]	Pilot	54	650
11	65a1284eb5ab415a36a734b3	f	["grape", "carrot"]	Nimue	<unset>	540
12	65a173c5c7dd9e6803b776af	m	["grape", "watermelon"]	Dunx	165	704
13	65a1753fc7dd9e6803b776b1	m	["grape"]	Barney	<unset>	340

### Практическое задание 3.3.2:

Для самки единорога Айна внести изменения в БД: теперь ее вес 800, она убила 51 вампира.

Проверить содержимое коллекции *unicorns*.



The screenshot shows the MongoDB Playground interface. Two commands have been executed: `db.unicorns.updateOne({name: "Ayna"}, {$set: {weight: 800, vampires: 51}})` and `db.unicorns.find({name: "Ayna"})`. Below the commands, a table displays the updated data for the unicorn named Ayna. The table has 1 row with fields: `_id`, `gender`, `loves`, `name`, `vampires`, and `weight`.

	<code>_id</code>	<code>gender</code>	<code>loves</code>	<code>name</code>	<code>vampires</code>	<code>weight</code>
1	65a1284db5ab415a36a734ae	f	["strawberry", "lemon"]	Ayna	51	800

### Практическое задание 3.3.3:

1. Для самца единорога Raleigh внести изменения в БД: теперь он любит рэббул.

2. Проверить содержимое коллекции *unicorns*.

Playground

```

1 ✓ db.unicorns.updateOne({name : "Raleigh"}, {$set : {"loves" : ["redbull"]}})
2 ✓ db.unicorns.find({name : "Raleigh"})

```

Output test.unicorns 2 × test.unicorns

1 row

	_id	gender	loves	name	vampires	weight
1	65a1284db5ab415a36a734b0	m	["redbull"]	Raleigh	2	421

### Практическое задание 3.3.4:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.
2. Проверить содержимое коллекции `unicorns`.

Playground

```

1 ✓ db.unicorns.updateMany({gender : "m"}, {$inc : {vampires : 5}})
2 ✓ db.unicorns.find({gender : "m"}, {_id:0, name: 1, vampires: 1})

```

Output test.unicorns 2 × test.unicorns ×

8 rows

	name	vampires
1	Horny	68
2	Unicrom	187
3	Rooodoodles	104
4	Kenny	44
5	Raleigh	7
6	Pilot	59
7	Dunx	170
8	Barney	5

### Практическое задание 3.3.5:

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.
2. Проверить содержимое коллекции `towns`.

Playground

```

1 db.towns.updateOne({name : "Portland"}, {$unset: {"mayor.party" : 1}})
2 db.towns.find({name : "Portland"})

```

Output test.towns test.unicorns

	id	famous_for	last_sensus	mayor	name	populatiuon
1	71f0c7dd9e6803b776ad	["beer", "food"]	2009-07-20T00:00:00.000Z	{"name": "Sam Adams"}	Portland	528000

### Практическое задание 3.3.6:

1. Изменить информацию о самце единорога *Pilot*: теперь он любит и шоколад.
2. Проверить содержимое коллекции *unicorns*.

Playground

```

1 db.unicorns.updateOne({name : "Pilot"}, {$push: {loves: "chocolate"}})
2 db.unicorns.find({name : "Pilot"})

```

Output test.unicorns test.unicorns 2

	_id	gender	loves	name	vampires	weight
1	65a1284db5ab415a36a734b2	m	["apple", "watermelon", "chocolate"]	Pilot	59	650

### Практическое задание 3.3.7:

1. Изменить информацию о самке единорога *Aurora*: теперь она любит еще и сахар, и лимоны.
2. Проверить содержимое коллекции *unicorns*.

Playground

```

1 db.unicorns.updateOne({name : "Aurora"}, {$addToSet: {loves: {$each: ["sugar", "lemon"]}})
2 db.unicorns.find({name : "Aurora"})

```

Output test.unicorns test.unicorns 2

	_id	gender	loves	name	vampires	weight
1	65a1284cb5ab415a36a734aa	f	["carrot", "grape", "sugar", "lemon"]	Aurora	43	450

### Практическое задание 3.4.1:

1. *Создайте коллекцию towns, включающую следующие документы:*

```
{name: "Punxsutawney ",
popujatiuon: 6200,
last_sensus: ISODate("2008-01-31"),
famous_for: ["phil the groundhog"],
mayor: {
  name: "Jim Wehrle"
}},

{name: "New York",
popujatiuon: 22200000,
last_sensus: ISODate("2009-07-31"),
famous_for: ["status of liberty", "food"],
mayor: {
  name: "Michael Bloomberg",
  party: "I"}}

{name: "Portland",
popujatiuon: 528000,
last_sensus: ISODate("2009-07-20"),
famous_for: ["beer", "food"],
mayor: {
  name: "Sam Adams",
  party: "D"}}
```

```
db.towns.insertMany([
  {name: "Punxsutawney ",
    popujatiuon: 6200,
    last_sensus: ISODate("2008-01-31"),
    famous_for: ["phil the groundhog"],
    mayor: {
      name: "Jim Wehrle"
    }
  },
  {name: "New York",
    popujatiuon: 22200000,
    last_sensus: ISODate("2009-07-31"),
    famous_for: ["status of liberty", "food"],
    mayor: {
      name: "Michael Bloomberg",
      party: "I"
    }
  },
  {name: "Portland",
    popujatiuon: 528000,
    last_sensus: ISODate("2009-07-20"),
    famous_for: ["beer", "food"],
    mayor: {
      name: "Sam Adams",
      party: "D"
    }
  }
]);
```

2. *Удалите документы с беспартийными мэрами.*

```
db.towns.deleteMany({'mayor.party': {'$exists': 0}})
```

3. *Проверьте содержание коллекции.*

lorer

console x

1 db.towns.find()

test console

Output test.towns Result 66

learn	_id	famous_for	last_sensus	mayor	name	popujatiuon
1	65a178fec7dd9e6803b776b6	["status of liberty", "food"]	2009-07-31T00:00:00.000Z	{"name": "Michael Bloomberg", "party": "I"}	New York	22200000
2	65a178fec7dd9e6803b776b7	["beer", "food"]	2009-07-20T00:00:00.000Z	{"name": "Sam Adams", "party": "D"}	Portland	528000

4. Очистите коллекцию.

1 db.towns.deleteMany({});

Output test.towns Result 70 x

	acknowledged	deletedCount
1	• true	2

5. Просмотрите список доступных коллекций.

1 show collections

2

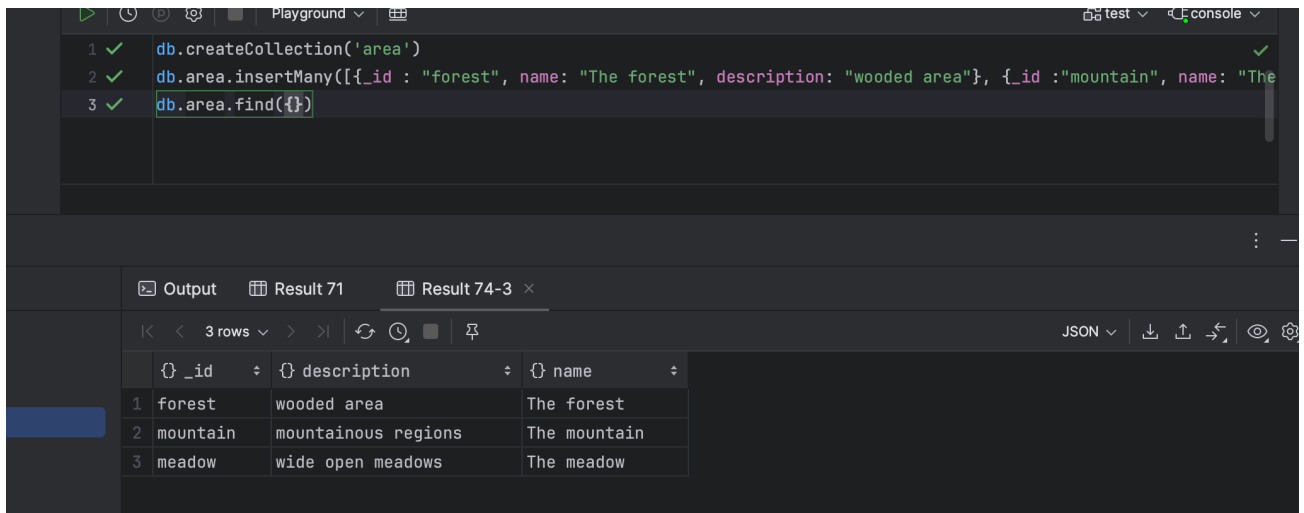
Output Result 71 x Result 70

	badge	name
1		towns
2		unicorns

### Практическое задание 4.1.1:



1. Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.



```
1 db.createCollection('area')
2 db.area.insertMany([{_id: "forest", name: "The forest", description: "wooded area"}, {_id: "mountain", name: "The mountain", description: "mountainous regions"}, {_id: "meadow", name: "The meadow", description: "wide open meadows"}])
3 db.area.find({})
```

	_id	description	name
1	forest	wooded area	The forest
2	mountain	mountainous regions	The mountain
3	meadow	wide open meadows	The meadow

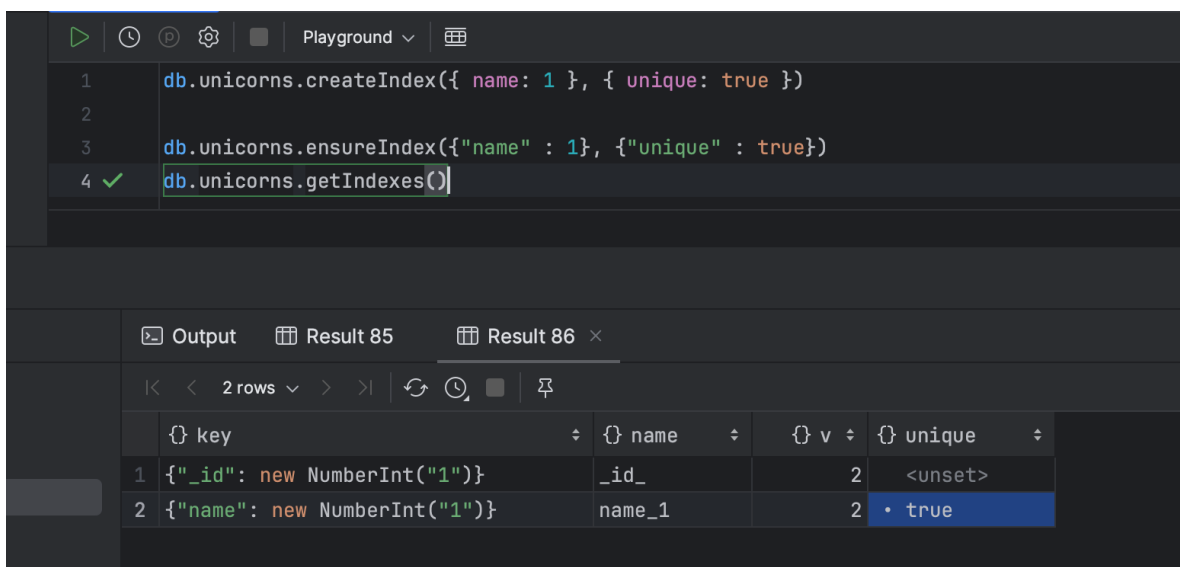
2. Включите для нескольких единорогов в документы ссылку на зону обитания, используя второй способ автоматического связывания.

```
3. db.unicorns.updateOne({_id: ObjectId("65a1753fc7dd9e6803b776b1")}, {$set: {location: {$ref: "area", $id: "mountain"}}})
```

```
4. db.unicorns.updateOne({_id: ObjectId("65a1284cb5ab415a36a734ac")}, {$set: {location: {$ref: "area", $id: "forest"}}})
```

### Практическое задание 4.2.1:

1. Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом unique.

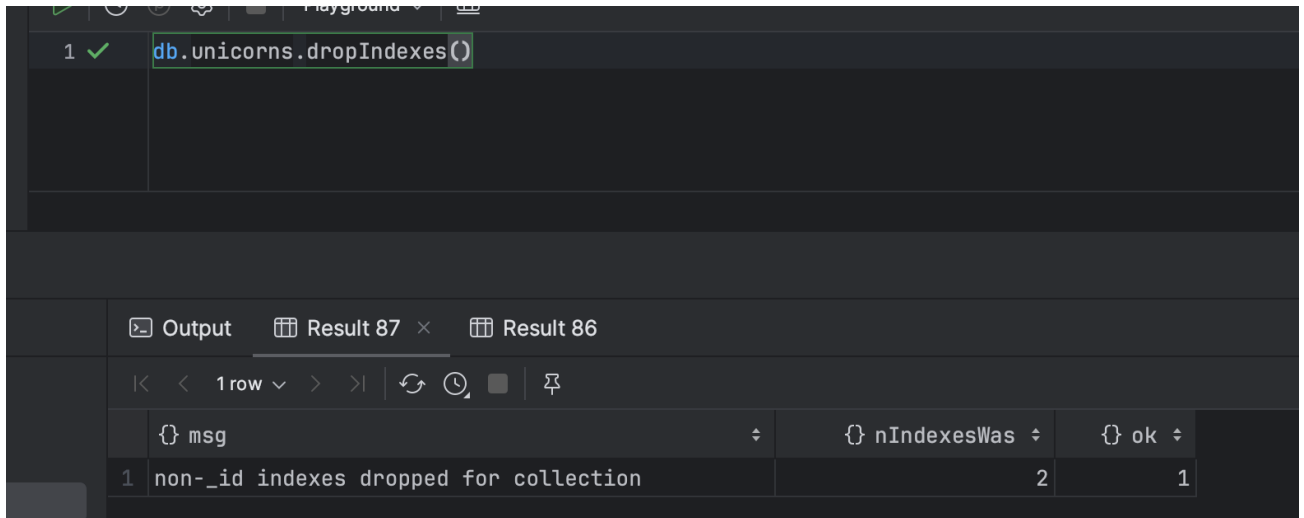


```
1 db.unicorns.createIndex({ name: 1 }, { unique: true })
2
3 db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})
4 db.unicorns.getIndexes()
```

	key	name	v	unique
1	{"_id": new NumberInt("1")}	_id_	2	<unset>
2	{"name": new NumberInt("1")}	name_1	2	• true

### Практическое задание 4.3.1:

1. Получите информацию о всех индексах коллекции `unicorns`.
2. Удалите все индексы, кроме индекса для идентификатора.
3. Попробуйте удалить индекс для идентификатора.



### Практическое задание 4.4.1:

1. Создайте объемную коллекцию `numbers`, задействовав курсор:

```
for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
```

3. Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`)
4. Создайте индекс для ключа `value`.
5. Получите информацию о всех индексах коллекции `numbers`.
6. Выполните запрос 2.
7. Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?
8. Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос:

какой запрос более эффективен?

```
for(i = 0; i < 100000; i++){db.numbers.insertOne({value: i})}
```

Playground

```
1 ✓ db.numbers.find().sort({value:-1}).limit(4)
```

Output Result 87 test.numbers ×

4 rows

	{_id}	{value}
1	65a18f5e13dae51eb68c17bb	99999
2	65a18f5e13dae51eb68c17ba	99998
3	65a18f5e13dae51eb68c17b9	99997
4	65a18f5e13dae51eb68c17b8	99996

Без индекса:

Playground

```
1 ✓ db.numbers.find().sort({value: -1}).limit(4).explain('executionStats').executionStats.executionTimeMillis
```

Output Result 90 Result 95 ×

1 row

	{result}
1	56

С индексом:

Playground

```
1 ✓ db.numbers.createIndex({value: -1})
2 ✓ db.numbers.find().sort({value: -1}).limit(4).explain('executionStats').executionStats.executionTimeMillis
3 |
```

Output Result 96 Result 96-2 ×

1 row

	{result}
1	3

Можно сделать вывод, что с индексом быстрее

Вывод:

В ходе лабораторной работы были использованы различные методы и освоены на практике NoSQL БД MongoDB.