

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Работа с БД в СУБД MongoDB»

по дисциплине «**Проектирование и реализация баз данных**»

Автор: Афолина Н.Р.

Факультет: ИКТ

Группа: К32421

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы	3
Практическое задание	3
Вывод.....	22

Цель работы

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.

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

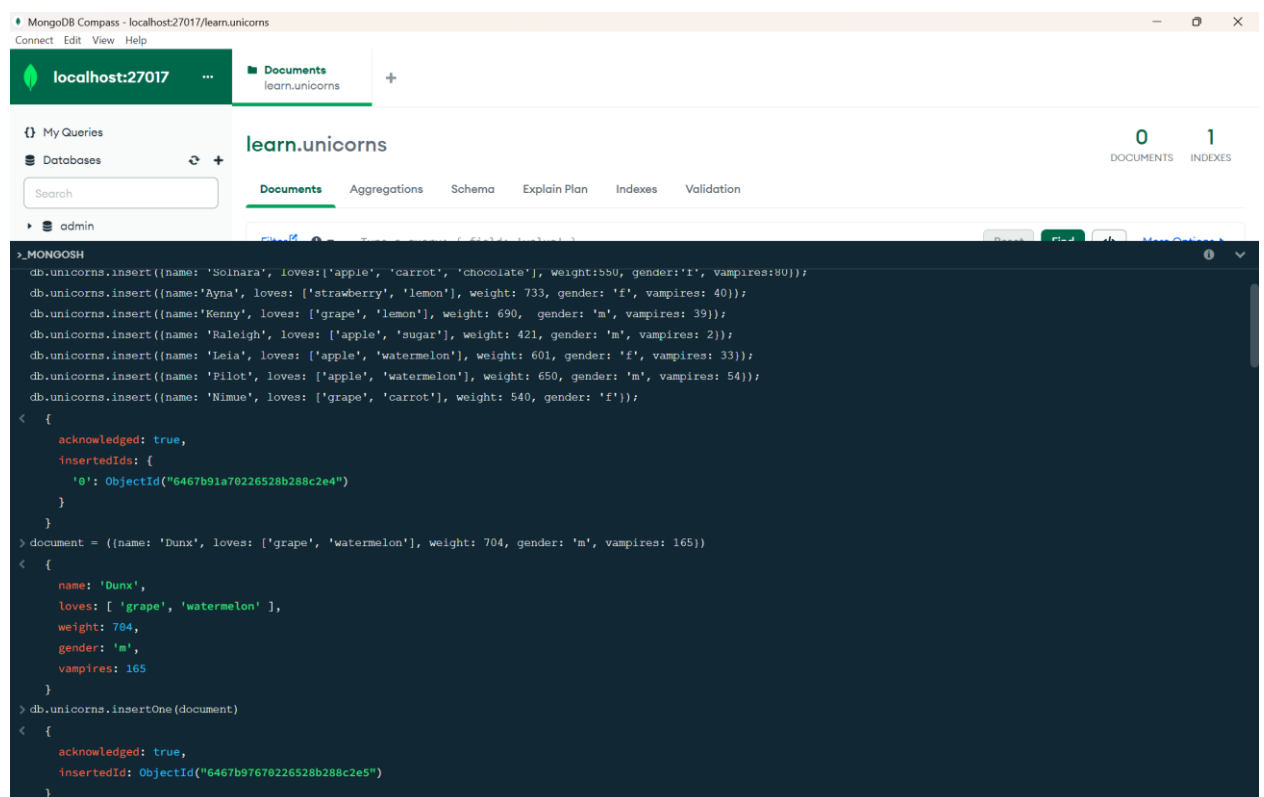
Практическое задание

8.1 CRUD-ОПЕРАЦИИ В СУБД MONGODB. ВСТАВКА ДАННЫХ. ВЫБОРКА ДАННЫХ

1. ВСТАВКА ДОКУМЕНТОВ В КОЛЛЕКЦИЮ

Практическое задание 8.1.1:

1. *Создайте базу данных learn.*
2. *Заполните коллекцию единорогов unicorns:*
3. *Используя второй способ, вставьте в коллекцию единорогов документ:*
4. *Проверьте содержимое коллекции с помощью метода find.*



The screenshot shows the MongoDB Compass interface for the 'learn.unicorns' database. The 'Documents' tab is active, showing a list of documents. Below the interface, a terminal window displays the following commands and their output:

```
> use learn
> db.unicorns.insert((name: 'Soinara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80));
db.unicorns.insert((name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40));
db.unicorns.insert((name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39));
db.unicorns.insert((name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2));
db.unicorns.insert((name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33));
db.unicorns.insert((name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54));
db.unicorns.insert((name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'));
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6467b91a70226528b288c2e4")
  }
}
> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
> db.unicorns.insertOne(document)
{
  acknowledged: true,
  insertedId: ObjectId("6467b97670226528b288c2e5")
}
```

Рисунок 1 – Пункты 1-3

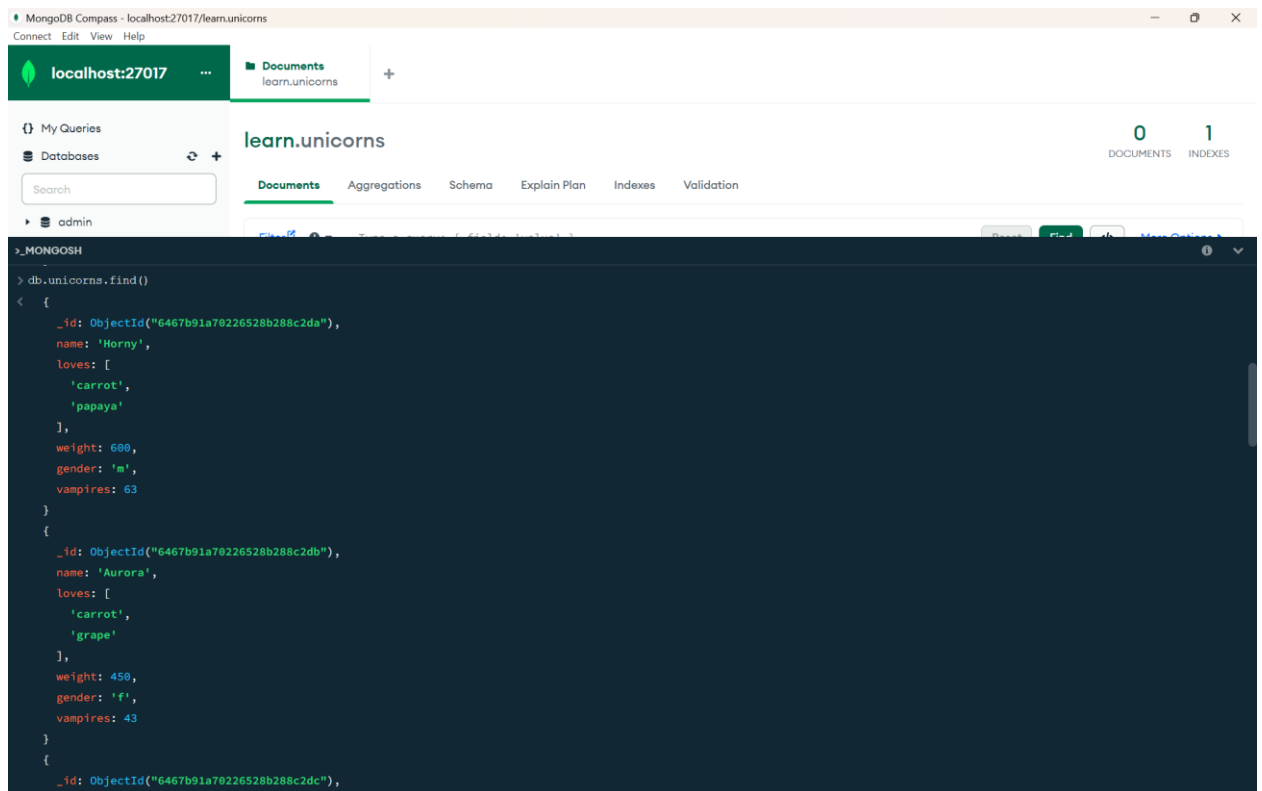


Рисунок 2 – Пункт 4

2. ВЫБОРКА ДАННЫХ ИЗ БД

Практическое задание 8.1.2:

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

Запрос:

```
db.unicorns.find({gender: 'm'}, {_id: 0}).sort({name: 1}).limit(3)
```

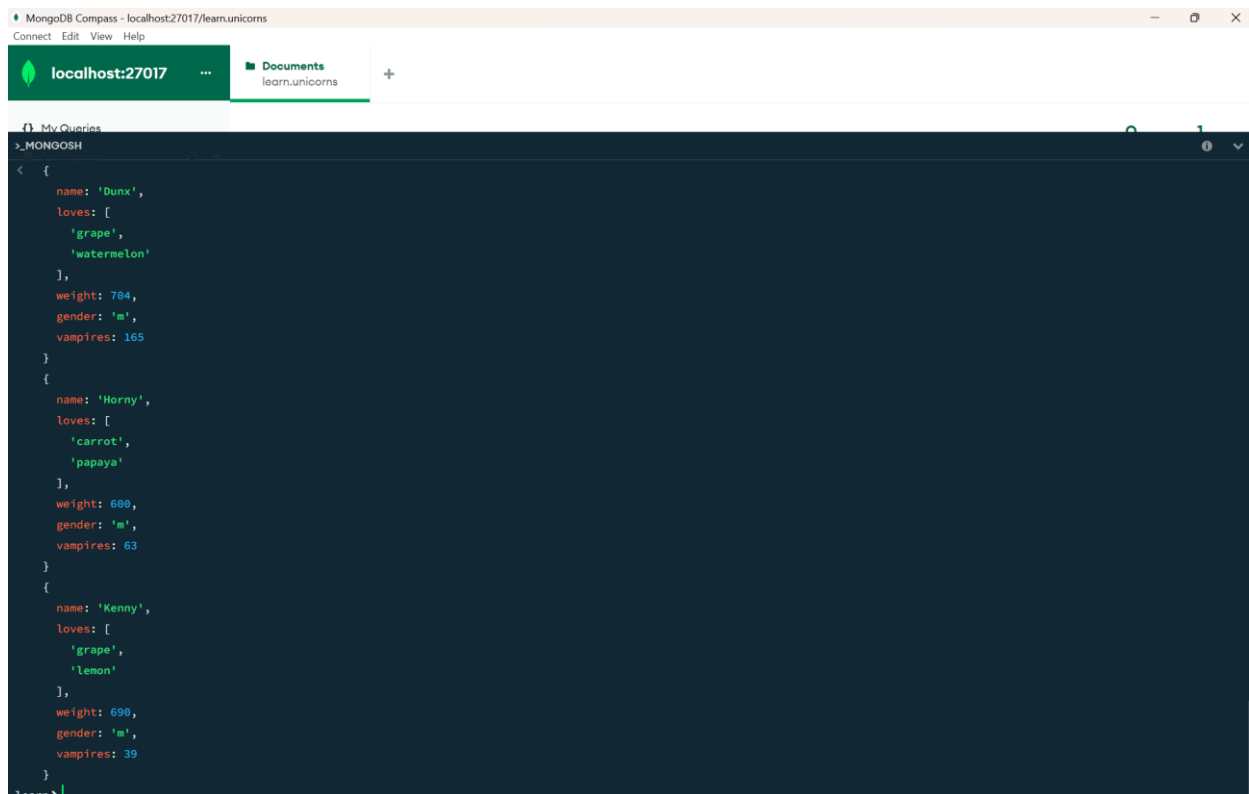


Рисунок 3 – Пункт 1 для самцов единорогов

Запрос:

```
db.unicorns.find({gender: "f"}, {_id: 0}).sort({name: 1}).limit(3)
```

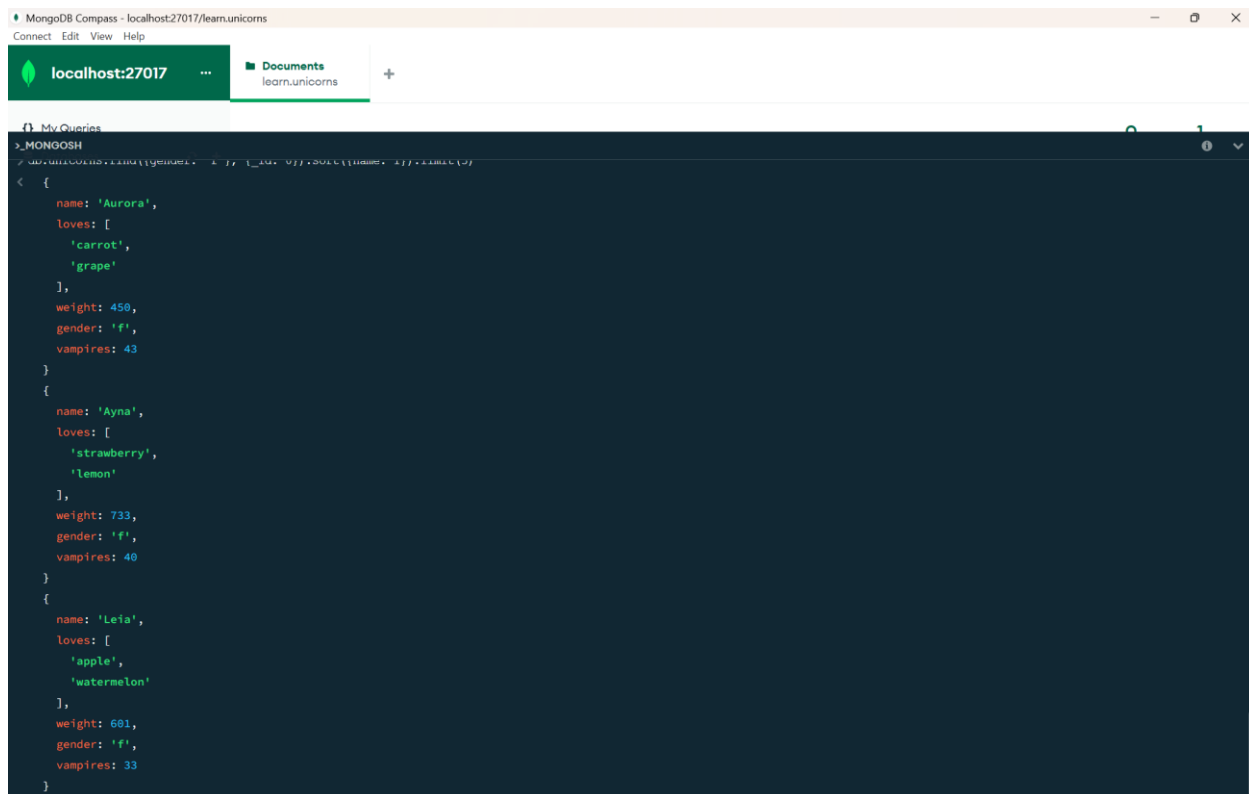


Рисунок 4 – Пункт 1 для самок единорогов

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

Запрос:

```
db.unicorns.findOne({gender: 'f', loves: 'carrot'})
```

ИЛИ

```
db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
```

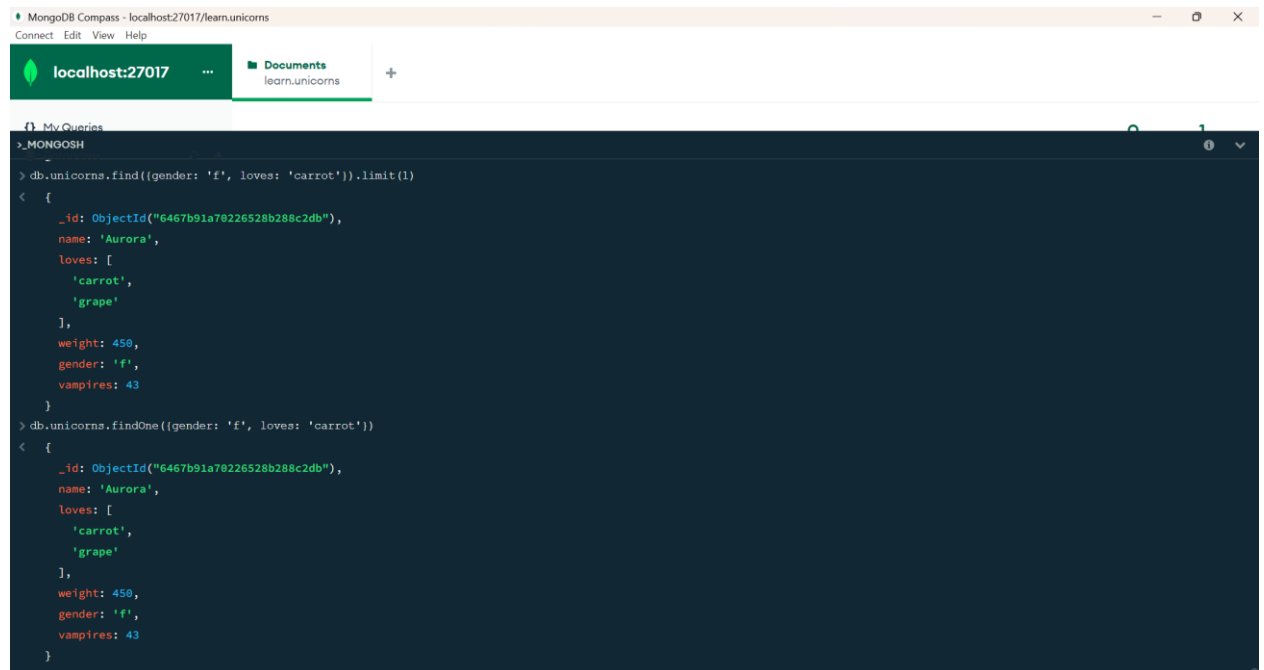


Рисунок 5 – Пункт 2

Практическое задание 8.1.3:

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

Запрос:

```
db.unicorns.find({gender: 'm'}, {_id: 0, loves: 0, gender: 0}).sort({name: 1}).limit(3)
```

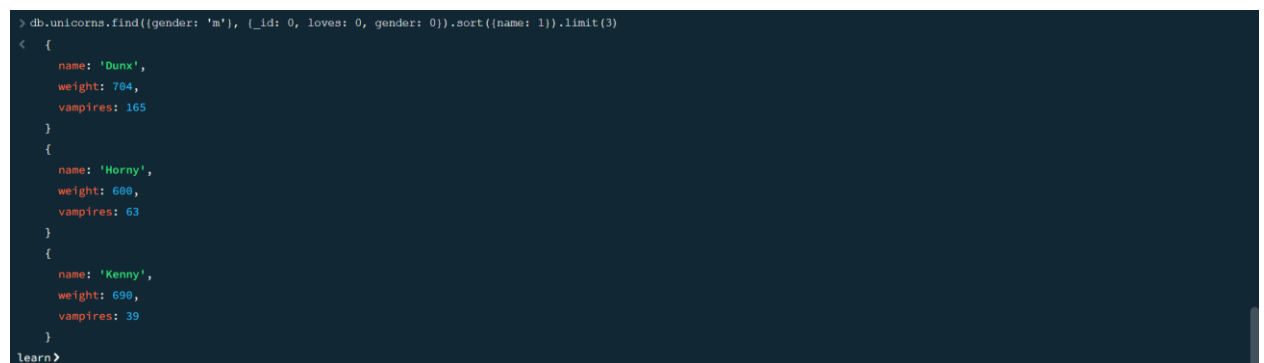


Рисунок 6 – Модифицированный запрос для задания 8.1.2, пункта 1

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

Запрос:

```
db.unicorns.find({gender: 'm'}, {_id: 0, name: 1}).sort({$natural: -1})
```

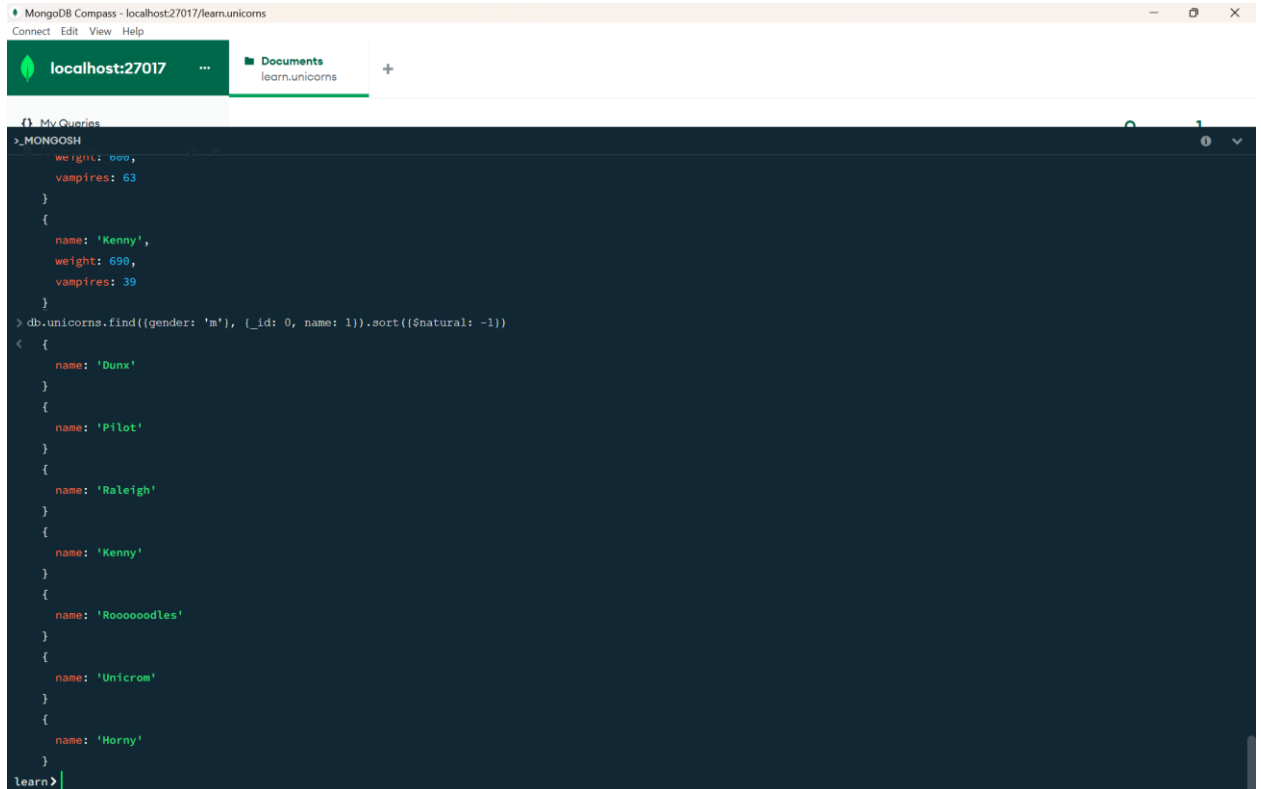


Рисунок 7 – Задание 8.1.4

Практическое задание 8.1.5:

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

Запрос:

```
db.unicorns.find({}, {_id: 0, loves: { $slice: 1 }})
```

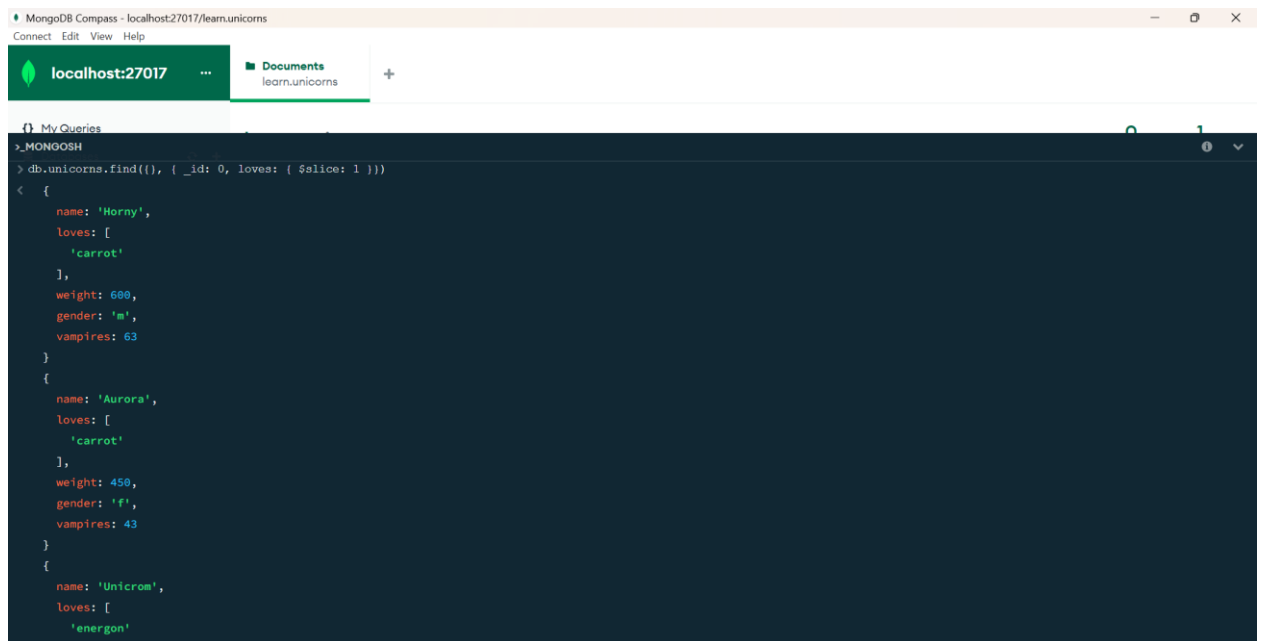


Рисунок 8 – Задание 8.1.5

3. ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Практическое задание 8.1.6:

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

Запрос:

```
db.unicorns.find({gender: "f", weight: { $gte: 500, $lte: 700 } },{_id: 0, name: 1, weight: 1})
```

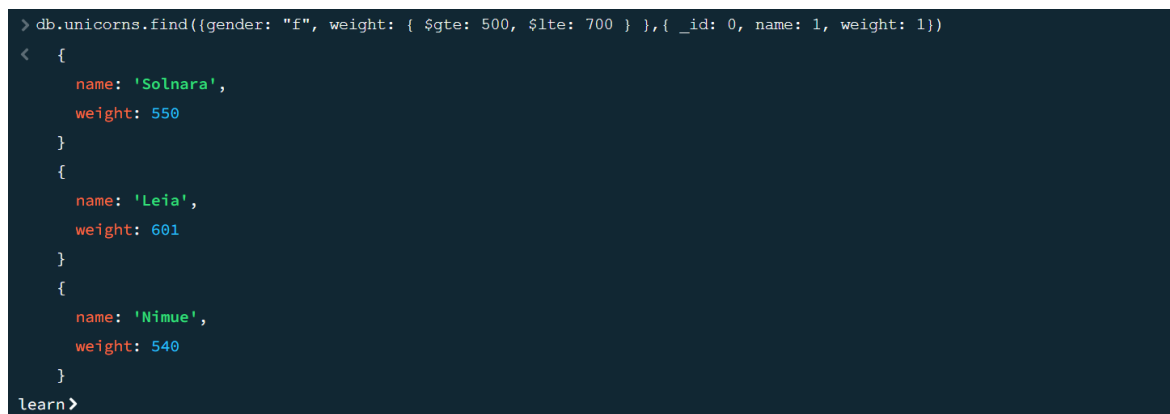


Рисунок 9 – Задание 8.1.6

Практическое задание 8.1.7:

Вывести список самцов единорогов весом от полутонны и предпочитающих grape и lemon, исключив вывод идентификатора.

Запрос:

```
db.unicorns.find({ gender: 'm', weight: { $gte: 500 }, loves: { $all: ["grape", "lemon"] } }, { _id: 0})
```



```

> db.unicorns.find({ gender: 'm', weight: { $gte: 500 }, loves: { $all: ["grape", "lemon"] } }, { _id: 0})
< {
  name: 'Kenny',
  loves: [
    'grape',
    'lemon'
  ],
  weight: 690,
  gender: 'm',
  vampires: 39
}
learn>

```

Рисунок 10 – Задание 8.1.7

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

Запрос:

```
db.unicorns.find({vampires: {$exists:false}})
```

```

> db.unicorns.find({vampires: {$exists:false}})
< {
  _id: ObjectId("6467b91a70226528b288c2e4"),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
learn>

```

Рисунок 11 – Задание 8.1.8

Практическое задание 8.1.9:

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

Запрос:

```
db.unicorns.find({gender: "m"}, {_id:0, name: 1, loves: { $slice: 1
}}).sort({name: 1})
```

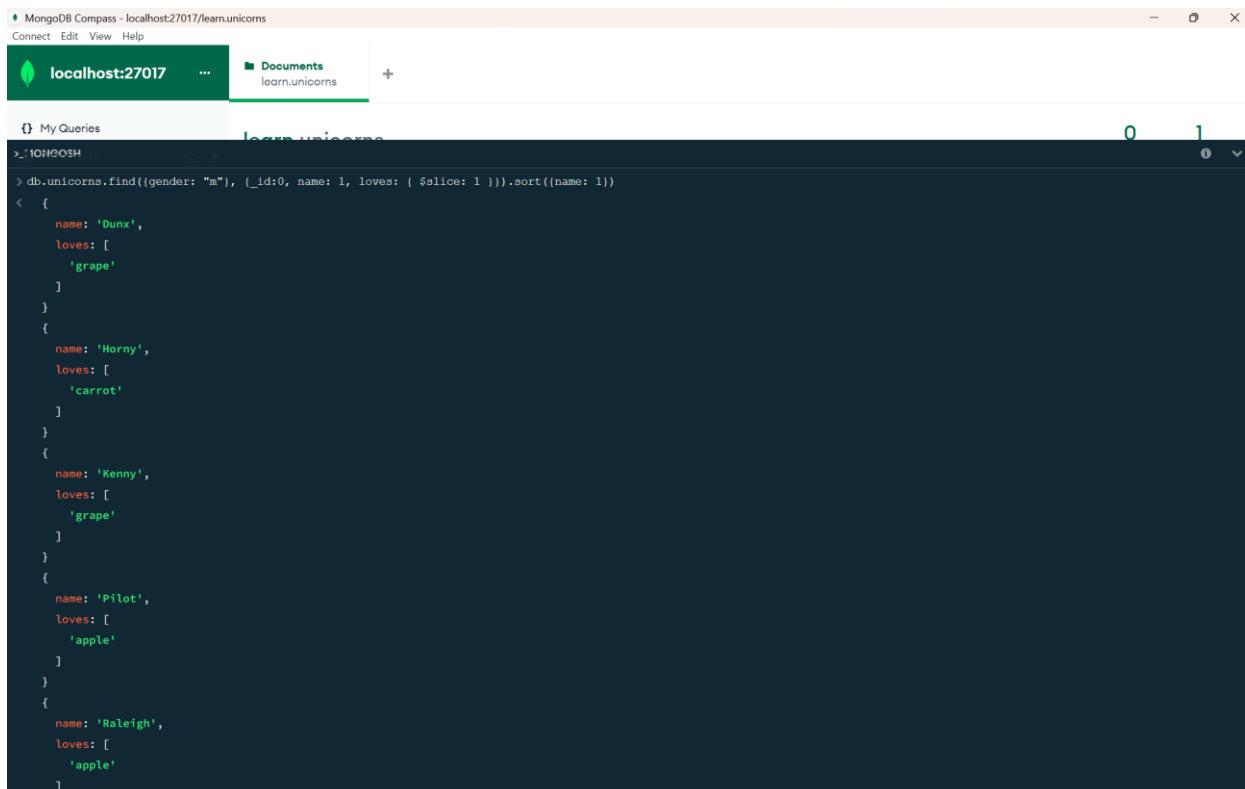


Рисунок 12 – Задание 8.1.9

8.2 ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB.

ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ.
АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ

Практическое задание 8.2.1:

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

Запрос:

```
db.towns.find({"mayor.party": "I"}, {_id: 0, name: 1, mayor: 1})
```

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

Запрос:

```
db.towns.find({"mayor.party": {$exists: false}}, {_id: 0, name: 1, mayor: 1})
```

```

> db.towns.find({"mayor.party": "I"}, {_id: 0, name: 1, mayor: 1})
< {
  name: 'New York',
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
> db.towns.find({"mayor.party": {$exists: false}}, {_id: 0, name: 1, mayor: 1})
< {
  name: 'Punxsutawney',
  mayor: {
    name: 'Jim Wehrle'
  }
}
learn>

```

Рисунок 13 – Задание 8.2.1 пункты 2, 3

Практическое задание 8.2.2:

Сформировать функцию для вывода списка самцов единорогов.

```

> fn = function() {return this.gender=="m"}
< [Function: fn]
> db.unicorns.find(fn).ToArray()
~ db.unicorns.find(fn).ToArray()

```

Рисунок 14 – Функция

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

Вывести результат, используя forEach.

Функция создавалась, но не получилось ее запустить ни одним способом, так как код не загружался.

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

Запрос:

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

```

> db.unicorns.find({ gender: "f", weight: { $gte: 500, $lte: 600 } }).count()
< 2
learn>

```

Рисунок 15 – задание 8.2.3

Практическое задание 8.2.4:

Вывести список предпочтений.

Запрос:

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

```
> db.unicorns.distinct("loves")
< [
  'apple',      'carrot',
  'chocolate', 'energon',
  'grape',      'lemon',
  'papaya',     'redbull',
  'strawberry', 'sugar',
  'watermelon'
]
```

Рисунок 16 – Задание 8.2.4

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов.

Запрос:

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

```
> db.unicorns.aggregate({$group: {_id: "$gender", count: { $sum: 1 }}})
< {
  _id: 'm',
  count: 7
}
{
  _id: 'f',
  count: 5
}
```

Рисунок 17 – Задание 8.2.5

Практическое задание 8.2.6:

1. Выполнить команду:

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

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

2. Проверить содержимое коллекции *unicorns*.

```

    gender: 'f'
  }
  {
    _id: ObjectId("6467b97670226528b288c2e5"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("6469e51ac2f7dab225267ef2"),
    name: 'Barny',
    loves: [
      'grape'
    ],
    weight: 340,
    gender: 'm'
  }
}
learn>

```

Рисунок 18 – Задание 8.2.6

Практическое задание 8.2.7:

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

Запрос:

```
db.unicorns.updateOne({ name: "Ayna", gender: "f" },{$set: {weight: 800,
vampires: 51}})
```

```

> db.unicorns.updateOne({ name: "Ayna", gender: "f" },{$set: {weight: 800, vampires: 51}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

```

Рисунок 19 – Успешное изменение

2. Проверить содержимое коллекции unicorns.

```

> db.unicorns.find((name: "Ayna"))
< {
  _id: ObjectId("6467b91a70226528b288c2df"),
  name: 'Ayna',
  loves: [
    'strawberry',
    'lemon'
  ],
  weight: 800,
  gender: 'f',
  vampires: 51
}
learn>

```

Рисунок 20 – Проверка изменений у единорога Айна

Практическое задание 8.2.8:

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

Запрос:

```
db.unicorns.updateOne({ name: "Raleigh", gender: "m" }, {$set: {loves: ["redbull"]}})
```

```
> db.unicorns.updateOne({ name: "Raleigh", gender: "m" }, {$set: {loves: ["redbull"]}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Рисунок 21 – Успешное изменение

2. Проверить содержимое коллекции *unicorns*.

```

}
> db.unicorns.find({name: "Raleigh"})
< {
  _id: ObjectId("6467b91a70226528b288c2e1"),
  name: 'Raleigh',
  loves: [
    'redbull'
  ],
  weight: 421,
  gender: 'm',
  vampires: 2
}
learn>
```

Рисунок 22 – Задание 8.2.8 пункт 2

Практическое задание 8.2.9:

1. Всем самцам единорогов увеличить количество убитых вампиров на 5.

Запрос:

```
db.unicorns.updateMany({gender: "m"}, {$inc: {vampires: 5}})
```

2. Проверить содержимое коллекции *unicorns*.

```

> db.unicorns.find({gender: "m"}, {vampires: 1});
{
  _id: ObjectId("6467b91a70226528b288c2da"),
  vampires: 63
}
{
  _id: ObjectId("6467b91a70226528b288c2dc"),
  vampires: 182
}
{
  _id: ObjectId("6467b91a70226528b288c2dd"),
  vampires: 99
}
{
  _id: ObjectId("6467b91a70226528b288c2e0"),
  vampires: 39
}
{
  _id: ObjectId("6467b91a70226528b288c2e1"),
  vampires: 2
}
{
  _id: ObjectId("6467b91a70226528b288c2e3"),
  vampires: 54
}
{
  _id: ObjectId("6467b97670226528b288c2e5")
}

```

Рисунок 23 – Количество убитых вампиров до изменений

```

> db.unicorns.find({gender: "m"}, {vampires: 1}).toArray()
[
  { _id: ObjectId("6467b91a70226528b288c2da"), vampires: 68 },
  { _id: ObjectId("6467b91a70226528b288c2dc"), vampires: 187 },
  { _id: ObjectId("6467b91a70226528b288c2dd"), vampires: 104 },
  { _id: ObjectId("6467b91a70226528b288c2e0"), vampires: 44 },
  { _id: ObjectId("6467b91a70226528b288c2e1"), vampires: 7 },
  { _id: ObjectId("6467b91a70226528b288c2e3"), vampires: 59 },
  { _id: ObjectId("6467b97670226528b288c2e5"), vampires: 170 },
  { _id: ObjectId("6469e51ac2f7dab225267ef2"), vampires: 5 }
]
learn>

```

Рисунок 24 – Количество убитых вампиров после изменений

Для удаления отдельного ключа используется оператор **\$unset**:

```
> db.users.update({name: "Tom"}, {$unset: {salary: 1}})
```

Если подобного ключа в документе не существует, то оператор не оказывает никакого влияния.

Можно удалять сразу несколько ключей:

```
> db.users.update({name: "Tom"}, {$unset: {salary: 1, age: 1}})
```

Практическое задание 8.2.10:

1. Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.

Запрос:

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

2. Проверить содержимое коллекции towns.

```
    name: 'Sam Adams',
    party: 'D'
  }
}
>> db.towns.update({name: "Portland"}, {$unset: {"mayor.party": 1}})
Error: clone(t=1){const r=t.loc||{};return e({loc:new Position("line"in r?r.line:this.loc.line,"column"in r?r.column:...<omitted>...)} could not be cloned.
> db.towns.updateOne({name: "Portland"}, {$unset: {"mayor.party": 1}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.towns.find({name: "Portland"});
< {
  _id: ObjectId("6469d1f370226528b28c2e8"),
  name: 'Portland',
  population: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams'
  }
}
```

Рисунок 25 – Задание 8.2.10

Практическое задание 8.2.11:

1. Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.\

Запрос:

```
db.unicorns.updateOne({name: "Pilot"}, {$push: {loves: "chocolate"}});
```

2. Проверить содержимое коллекции unicorns.

```
> db.unicorns.updateOne({name: "Pilot"}, {$push: {loves: "chocolate"}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.find({name: "Pilot"});
< {
  _id: ObjectId("6467b91a70226528b288c2e3"),
  name: 'Pilot',
  loves: [
    'apple',
    'watermelon',
    'chocolate'
  ],
  weight: 650,
  gender: 'm',
  vampires: 59
}
```

Рисунок 26 – Задание 8.2.11

Оператор **\$addToSet** подобно оператору **\$push** добавляет объекты в массив. Отличие состоит в том, что **\$addToSet** добавляет данные, если их еще нет в массиве:


```
> db.users.update({name : "Tom"}, {$addToSet: {languages: "russian"}})
```

Используя оператор **\$each**, можно **добавить сразу несколько значений**:

```
> db.users.update({name : "Tom"},  
  {$addToSet: {languages: {$each: ["russian", "spanish", "italian"]}}})
```

Практическое задание 8.2.12:

1. *Изменить информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны.*

Запрос:

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

2. *Проверить содержимое коллекции unicorns.*

```
> db.unicorns.updateOne({name: "Aurora" }, { $addToSet: {loves: {$each: ["sugar", "lemon"]}}})  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}  
> db.unicorns.find({name: "Aurora"})  
< {  
  _id: ObjectId("6467b91a70226528b288c2db"),  
  name: 'Aurora',  
  loves: [  
    'carrot',  
    'grape',  
    'sugar',  
    'lemon'  
  ],  
  weight: 450,  
  gender: 'f',  
  vampires: 43  
}
```

Рисунок 27 – Задание 8.2.12

Практическое задание 8.2.13:

4. *Создайте коллекцию towns (towns2)*
5. *Удалите документы с беспартийными мэрами.*

Запрос:

```
db.towns2.deleteMany({"mayor.party": { $exists: false }})
```

6. *Проверьте содержание коллекции.*

```

> db.towns2.count()
< 3
> db.towns2.deleteMany({"mayor.party": { $exists: false }})
< {
  acknowledged: true,
  deletedCount: 1
}
> db.towns2.count()
< 2
> db.towns2.find()
< {
  _id: ObjectId("6469f49ec2f7dab225267ef4"),
  name: 'New York',
  population: 22200000,
  last_sensus: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'statue of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}

```

Рисунок 28 – Задание 8.2.13

7. *Очистите коллекцию.*

Запрос:

```
db.towns2.deleteMany({});
```

8. *Просмотрите список доступных коллекций.*

```

> db.towns2.deleteMany({});
< {
  acknowledged: true,
  deletedCount: 2
}
> db.towns2.find()
<
> show collections
< towns
  towns2
  unicorns
> db.towns2.drop()
< true
> show collections
< towns
  unicorns

```

Рисунок 29 – Задание 8.2.13 пункты 7, 8

3. ССЫЛКИ И РАБОТА С ИНДЕКСАМИ В БАЗЕ ДАННЫХ MONGODB

Практическое задание 8.3.1:

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

```

    _id: "zone2",
    name: "Unicorn Habitat Zone 2",
    description: "This is the second habitat zone for unicorns"
  }
});
< {
  acknowledged: true,
  insertedIds: {
    '0': 'zone1',
    '1': 'zone2'
  }
}
> db.habitats.find()
< {
  _id: 'zone1',
  name: 'Unicorn Habitat Zone 1',
  description: 'This is the first habitat zone for unicorns'
}
{
  _id: 'zone2',
  name: 'Unicorn Habitat Zone 2',
  description: 'This is the second habitat zone for unicorns'
}
learn>

```

Рисунок 30 – Создание двух зон обитания единорогов

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

Запрос:

```

db.unicorns.updateOne({ name: "Horny" },{$set: {habitat: {$ref: "habitats", $id: "zone1"}}});

```

```

> db.unicorns.updateOne({ name: "Horny" },{$set: {habitat: {$ref: "habitats", $id: "zone1"}}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.updateOne({ name: "Aurora" },{$set: {habitat: {$ref: "habitats", $id: "zone1"}}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.unicorns.updateOne({ name: "Unicrom" },{$set: {habitat: {$ref: "habitats", $id: "zone2"}}});
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

Рисунок 31 – Добавление зон обитания для трех единорогов

3) Проверьте содержание коллекции единорогов.

```
db.unicorns.find({}, {_id: 0, name: 1, habitat: 1});
{
  name: 'Horny',
  habitat: DBRef("habitats", 'zone1')
}
{
  name: 'Aurora',
  habitat: DBRef("habitats", 'zone1')
}
{
  name: 'Unicrom',
  habitat: DBRef("habitats", 'zone2')
}
{
  name: 'Roooooodles'
}
{
  name: 'Solnara'
}
{
  name: 'Ayna'
}
{
  name: 'Kenny'
```

Рисунок 32 – Содержание коллекции единорогов после изменений

Практическое задание 8.3.2:

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

```
> db.unicorns.ensureIndex({"name" : 1}, {"unique" : true});
< [ 'name_1' ]
> db.unicorns.ensureIndex({"gender" : 1}, {"unique" : true});
✖ ► MongoServerError: Index build failed: 2446ea13-90b2-4253-b11b-363d7289fe9a: Colle
learn >
```

Рисунок 33 – Создание уникального индекса по имени и неудачная попытка создания уникального индекса по полу

Практическое задание 8.3.3:

Получите информацию о всех индексах коллекции *unicorns*.

Удалите все индексы, кроме индекса для идентификатора.

Попытайтесь удалить индекс для идентификатора.

```

> db.unicorns.getIndexes()
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
> db.unicorns.dropIndexes()
< {
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
> db.unicorns.dropIndex("_id");
< {
  ok: 0,
  errmsg: 'index not found with name [_id]',
  code: 27,
  codeName: 'IndexNotFound'
}
> db.unicorns.dropIndex("_id_");
✖ ► MongoServerError: cannot drop _id index

```

Рисунок 34 – Индексы в коллекции единорогов и их удаление

4. ПЛАН ЗАПРОСА

С помощью метода `explain()` можно получить информацию о выполнении запроса. Метод `explain()` возвращает JSON-структуру с планом выполнения запроса.

Для детализации плана запроса нужно указать параметр `"executionStats"` для метода `explain()`:

```
db.users.explain("executionStats").find({})
```

Практическое задание 8.3.4:

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

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

2. *Выберите последних четыре документа.*

3. *Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса? (по значению параметра `executionTimeMillis`)*

```

var query = db.numbers.find().sort({ _id: -1 }).limit(4);
var explainResult = query.explain("executionStats");
print("Execution time: " + explainResult.executionStats.executionTimeMillis +
" ms");

```

```

> var query = db.numbers.find().sort({ _id: -1 }).limit(4);
  var explainResult = query.explain("executionStats");
  print("Execution time: " + explainResult.executionStats.executionTimeMillis + " ms");
< 'Execution time: 40 ms'

```

Рисунок 35 – Пункт 3

4. *Создайте индекс для ключа `value`.*

5. *Получите информацию о всех индексах коллекции `numbers`.*

```
> db.numbers.getIndexes();
< [
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { value: 1 }, name: 'value_1' }
]
learn>
```

Рисунок 36 – Информация об индексах

6. *Выполните запрос 2.*
7. *Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?*

```
> var cursorWithIndex = db.numbers.find().sort({ _id: -1 }).limit(4);
> var explainResultIndex = cursorWithIndex.explain("executionStats");
print("Execution time: " + explainResultIndex.executionStats.executionTimeMillis + " ms");
< 'Execution time: 0 ms'
```

Рисунок 37 – Пункт 7

8. *Сравните время выполнения запросов с индексом и без. Дайте ответ на вопрос: какой запрос более эффективен?*

Наиболее эффективным оказался запрос с индексом, так как времени на его выполнение потребовалось меньше.

Вывод

В данной лабораторной работе я попрактиковалась в работе с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB.