Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Введение в СУБД MONGODB. Установка MONGODB. Начало работы с БД.Работа с БД в СУБД MongoDB»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Шестак Богдан Евгеньевич
Факультет прикладной информатики
Группа K3240
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2025
Преподаватель Говорова Марина Михайловна

Санкт-Петербург

2025/2026

1. Цель работы:

Овладеть практическими навыками установки СУБД MongoDB. 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.

2. Задание 1:

Скачали и установили monogodb с официального сайта, распаковали файлы и теперь проверяем работоспособность системы с помощью базового метода db.help()

```
> db.help()
< Database Class

  getMongo                  Returns the current database connection
  getName                   Returns the name of the DB
  getCollectionNames        Returns an array containing the names of all collections in the current database.
  getCollectionInfos        Returns an array of documents with collection information, i.e. collection name and options, for the
                            current database.
  runCommand                Runs an arbitrary command on the database.
  adminCommand              Runs an arbitrary command against the admin database.
  aggregate                 Runs a specified admin/diagnostic pipeline which does not require an underlying collection.
  getSiblingDB              Returns another database without modifying the db variable in the shell environment.
  getCollection             Returns a collection or a view object that is functionally equivalent to using the db.<collectionName>.
  dropDatabase              Removes the current database, deleting the associated data files.
  createUser                Creates a new user for the database on which the method is run. db.createUser() returns a duplicate
                            user error if the user already exists on the database.
  updateUser                Updates the user's profile on the database on which you run the method. An update to a field completely
                            replaces the previous field's values. This includes updates to the user's roles array.
  changeUserPassword        Updates a user's password. Run the method in the database where the user is defined, i.e. the database
                            you created the user.
  logout                    Ends the current authentication session. This function has no effect if the current session is not
                            authenticated.
  dropUser                  Removes the user from the current database.
```

Теперь пропишем db.stats(), которая выводит нулевые значения

```
> db.stats()
< {
    db: 'test',
    collections: Long('0'),
    views: Long('0'),
    objects: Long('0'),
    avgObjSize: 0,
    dataSize: 0,
    storageSize: 0,
    indexes: Long('0'),
    indexSize: 0,
    totalSize: 0,
    scaleFactor: Long('1'),
    fsUsedSize: 0,
    fsTotalSize: 0,
    ok: 1
  }
```

Далее создаем БД learn

```
> use learn
< switched to db learn
```

Далее добавим в нее коллекцию unicorns

```
> db.unicorns.insertOne({name: 'Aurora', gender: 'f', weight: 450})
< {
    acknowledged: true,
    insertedId: ObjectId('6837ad6192b5677c8588eb0a')
  }
```

Посмотрим список коллекция

```
> show collections
< unicorns
```

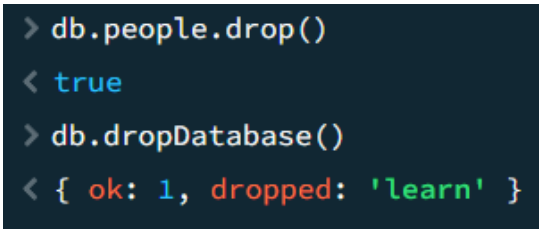
Переименуем коллекцию

```
> db.unicorns.renameCollection("people")
< { ok: 1 }
```

Посмотрим на статистику БД


},
sharded: false,
size: 65,
count: 1,
numOrphanDocs: 0,
storageSize: 20480,
totalIndexSize: 20480,
totalSize: 40960,
indexSizes: { _id_: 20480 },
avgObjSize: 65,
ns: 'learn.people',
nindexes: 1,
scaleFactor: 1
}

Удалим БД


> db.people.drop()
< true
> db.dropDatabase()
< { ok: 1, dropped: 'learn' }

3. Задание 2.1.1:

Добавляем в базу данных коллекции единорогов unicorns

```
> db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});
  db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});
  db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});
  db.unicorns.insert({name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99});
  db.unicorns.insert({name: 'Solnara', 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'});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6837b4b192b5677c8588eb15')
    }
  }
```

Добавляем коллекцию как документ

```
> document=({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 764, gener: 'm', vampires:165})
< {
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 764,
    gener: 'm',
    vampires: 165
  }
> db.unicorns.insert(document)
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6837b59592b5677c8588eb16')
    }
  }
```

Проверяем содержимое коллекции и обнаружим там запись про Dunx

```
{
  _id: ObjectId('6837b59592b5677c8588eb16'),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 764,
  gener: 'm',
  vampires: 165
}
```

4. Задание 2.2.1:

Выводим список самцов, сортируя их по имени

```
> db.unicorns.find({gender: 'm'}).sort({name: 1})
< {
    _id: ObjectId('6837b4b192b5677c8588eb0b'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb11'),
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb14'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb12'),
    name: 'Raleigh',
    loves: [
      'apple',
      'sugar'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

```
  {
    _id: ObjectId('6837b4b192b5677c8588eb0e'),
    name: 'Roooooodles',
    loves: [
      'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0d'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
```

Выводим списки самок, ограничив список самок первыми 3 особями и отсортировав список по имени.

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3)
< {
    _id: ObjectId('6837b4b192b5677c8588eb0c'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb10'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb13'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
```

Выводим список самок, которые любят carrot, ограничив список с помощью findOne и limit

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'})
< {
    _id: ObjectId('6837b4b192b5677c8588eb0c'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}).limit(1)
< {
    _id: ObjectId('6837b4b192b5677c8588eb0c'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

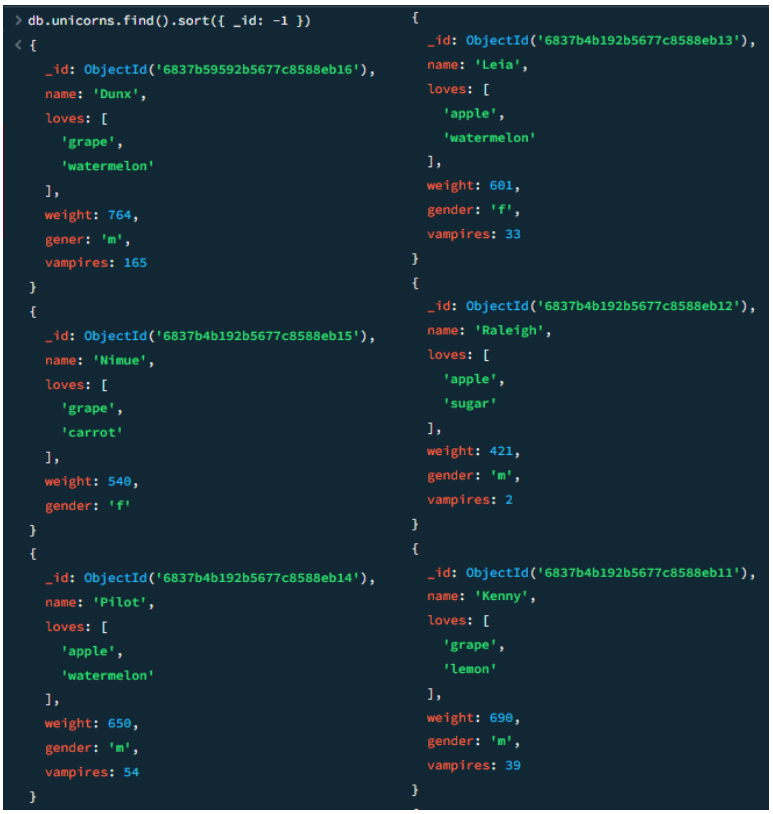
5. Задание 2.2.2:

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

```
> db.unicorns.find({ gender: 'm'}, { loves: 0, vampires: 0 })
< {
    _id: ObjectId('6837b4b192b5677c8588eb0b'),
    name: 'Horny',
    weight: 600,
    gender: 'm'
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0d'),
    name: 'Unicrom',
    weight: 984,
    gender: 'm'
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0e'),
    name: 'Roooooodles',
    weight: 575,
    gender: 'm'
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb11'),
    name: 'Kenny',
    weight: 690,
    gender: 'm'
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb12'),
    name: 'Raleigh',
    weight: 421,
    gender: 'm'
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb14'),
    name: 'Pilot',
    weight: 650,
    gender: 'm'
  }
```

6. Задание 2.2.3:

Выводим список единорогов в обратном порядке добавления

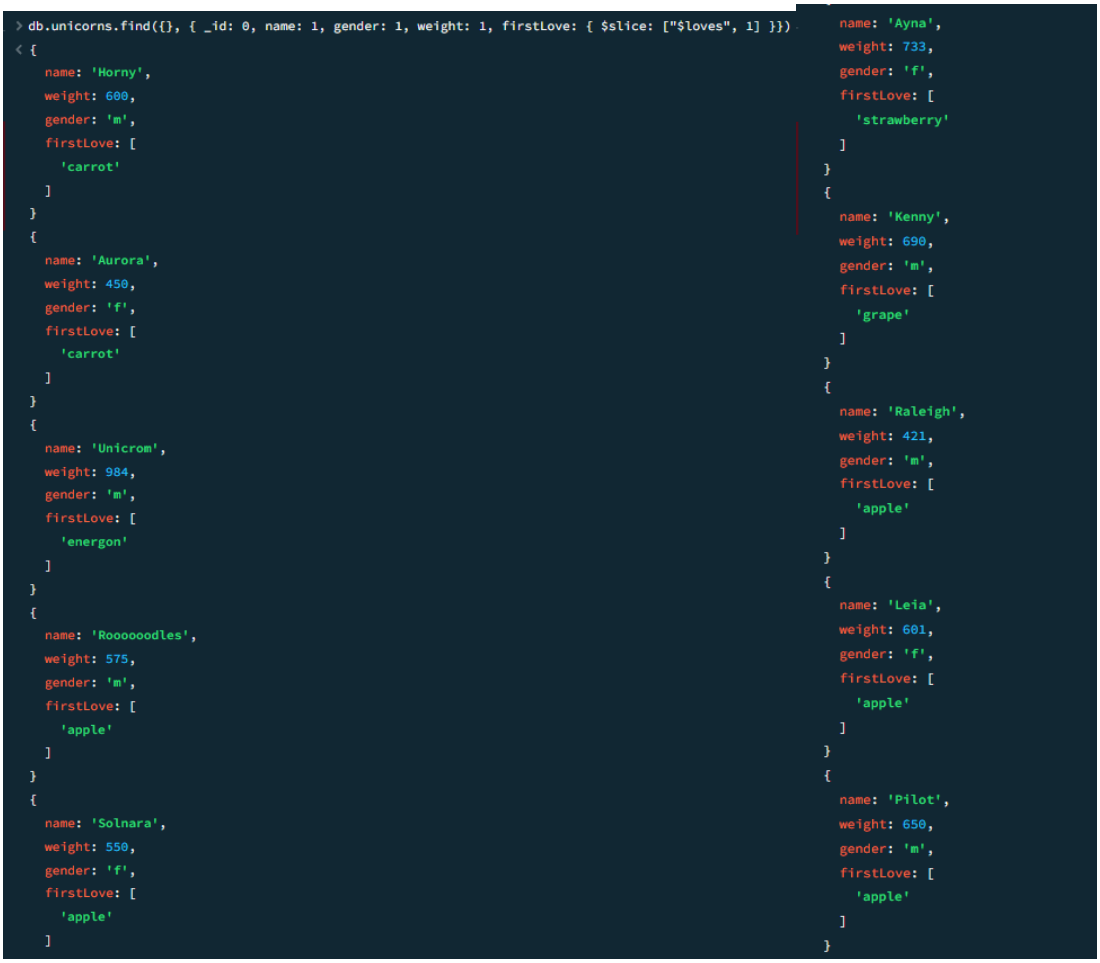


```
> db.unicorns.find().sort({ _id: -1 })
< {
    _id: ObjectId('6837b59592b5677c8588eb16'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 764,
    gener: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb15'),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb14'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 650,
    gender: 'm',
    vampires: 54
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb13'),
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
    vampires: 33
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb12'),
    name: 'Raleigh',
    loves: [
      'apple',
      'sugar'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb11'),
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
```

```
  {
    _id: ObjectId('6837b4b192b5677c8588eb10'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0f'),
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0e'),
    name: 'Roooooodles',
    loves: [
      'apple'
    ],
    weight: 575,
    gender: 'm',
    vampires: 99
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0d'),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0c'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0b'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
```

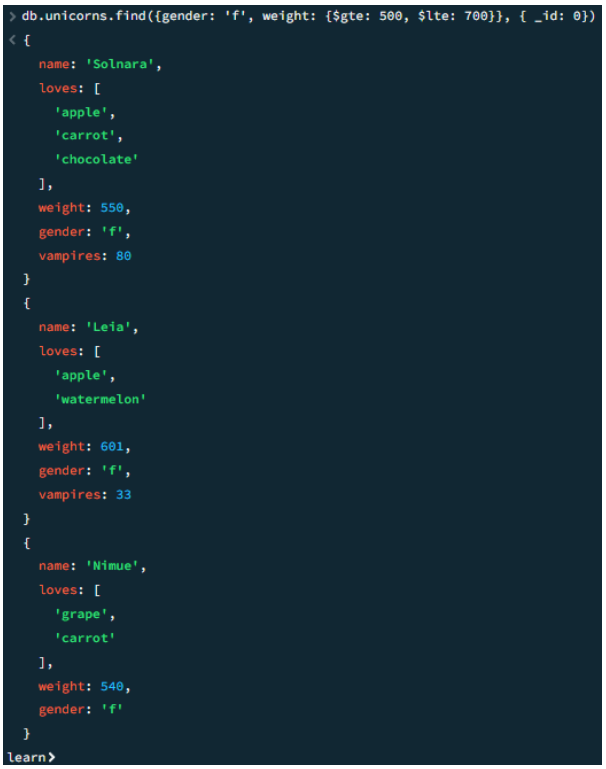
7. Задание 2.1.4:

Выведем список единорогов с названием первого любимого предпочтения, исключив идентификатор



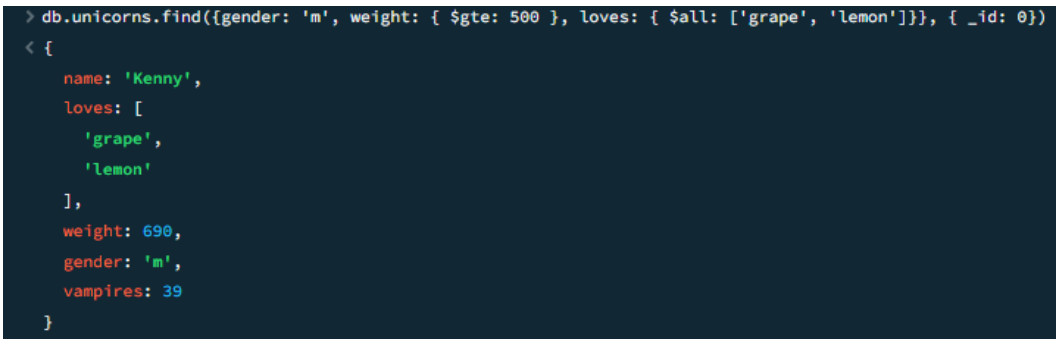
8. **Задание 2.3.1:**

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



9. **Задание 2.3.2:**

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



10. Задание 2.3.3:

Найти всех единорогов, не имеющих ключ vampires

```
> db.unicorns.find({ vampires: {$exists: false}}, { _id: 0})
< {
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
    gender: 'f'
  }
```

11. Задание 2.3.4:

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

```
> db.unicorns.aggregate([{$match: {gender: 'm'}},{ $project: {_id: 0, name: 1, firstLove: {$arrayElemAt: ["$loves", 0]}}}, {$sort: {name: 1}}])
< {
    name: 'Horny',
    firstLove: 'carrot'
  }
  {
    name: 'Kenny',
    firstLove: 'grape'
  }
  {
    name: 'Pilot',
    firstLove: 'apple'
  }
  {
    name: 'Raleigh',
    firstLove: 'apple'
  }
  {
    name: 'Roooooodles',
    firstLove: 'apple'
  }
  {
    name: 'Unicrom',
    firstLove: 'energon'
  }
```

12. Задание 3.1.1:

Создадим коллекцию towns

```
> db.towns.insertMany([{ name: "Punxsutawney ", populatiuon: 6200,
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6837cf3292b5677c8588eb17'),
      '1': ObjectId('6837cf3292b5677c8588eb18'),
      '2': ObjectId('6837cf3292b5677c8588eb19')
    }
  }
```

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

```
> db.towns.find({"mayor.party": "I"}, {_id: 0, name: 1, mayor: 1})
< {
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

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

```
> db.towns.find({"mayor.party": {$exists: false}}, {_id: 0, name: 1, mayor: 1})
< {
    name: 'Punxsutawney ',
    mayor: {
      name: 'Jim Wehrle'
    }
  }
```

13. Задание 3.1.2:

Сформировать функцию для вывода списка самцов единорогов.

```
> function getMaleUnicorns() {return db.unicorns.find({gender: 'm'}).sort({name: 1}).limit(2);}
< [Function: getMaleUnicorns]
```

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

```
> maleUnicornsCursor.forEach(function(unicorn)
< {
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    vampires: 63
  }
< {
    name: 'Kenny',
    loves: [ 'grape', 'lemon' ],
    weight: 690,
    vampires: 39
  }
```

14. Задание 3.2.1:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.countDocuments({gender: 'f', weight: { $gte: 500, $lte: 600}})
< 2
```

15. Задание 3.2.2:

Вывести список предпочтений.

```
> db.unicorns.aggregate([{$unwind: "$loves"}, {$group: {_id: "$loves"}}, {$project: {_id:0, food: "$_id"}}])
< {
    food: 'chocolate'
  }
  {
    food: 'lemon'
  }
  {
    food: 'sugar'
  }
  {
    food: 'watermelon'
  }
  {
    food: 'energon'
  }
  {
    food: 'carrot'
  }
  {
    food: 'strawberry'
  }
  {
    food: 'apple'
  }
  {
    food: 'grape'
  }
  {
    food: 'papaya'
  }
  {
    food: 'redbull'
  }
```

16. Задание 3.2.3

Подсчитать количество особей единорогов обоих полов.

```
> db.unicorns.aggregate([{$group: {_id: "$gender", count: {$sum: 1}}}, {$project: {gender: "_id", count: 1,_id: 0}}])
< {
    count: 6,
    gender: '_id'
  }
  {
    count: 1,
    gender: '_id'
  }
  {
    count: 5,
    gender: '_id'
  }
```

17. Задание 3.3.1:

Добавим нового единорога

```
> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
< {
    acknowledged: true,
    insertedId: ObjectId('6837d84192b5677c8588eb1a')
  }
```

Посмотрим на коллекцию unicorns (Barny появился)

```
> db.unicorns.find()
< {
    _id: ObjectId('6837b4b192b5677c8588eb0b'),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId('6837b4b192b5677c8588eb0c'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId('6837b59592b5677c8588eb16'),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 764,
    gener: 'm',
    vampires: 165
  }
  {
    _id: ObjectId('6837d84192b5677c8588eb1a'),
    name: 'Barny',
    loves: [
      'grape'
    ],
    weight: 340,
    gender: 'm'
  }
```

18. Задание 3.3.2:

Для самки единорога `Ayna` внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.

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

Посмотрим на обновленную запись единорога Ayna

```
> db.unicorns.find({ name: "Ayna"}).pretty()
< {
    _id: ObjectId('6837b4b192b5677c8588eb10'),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
```

19. Задание 3.3.3:

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

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

Посмотрим на обновлённую запись

```
> db.unicorns.find({ name: 'Raleigh'}).pretty()
< {
    _id: ObjectId('6837b4b192b5677c8588eb12'),
    name: 'Raleigh',
    loves: [
      'apple',
      'sugar',
      'RedBull'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

20. Задание 3.3.4:

Всем самцам единорогов увеличим количество убитых вампиров на 5.

```
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 7,
    modifiedCount: 7,
    upsertedCount: 0
  }
```

Посмотрим на обновленную запись

```
> db.unicorns.find({ gender: 'm'}, {name: 1, vampires: 1, _id: 0}).pretty()
< {
    name: 'Horny',
    vampires: 68
  }
  {
    name: 'Unicrom',
    vampires: 187
  }
  {
    name: 'Roooooodles',
    vampires: 104
  }
  {
    name: 'Kenny',
    vampires: 44
  }
  {
    name: 'Raleigh',
    vampires: 7
  }
  {
    name: 'Pilot',
    vampires: 59
  }
  {
    name: 'Barny',
    vampires: 5
  }
```

21. Задание 3.3.5:

Изменим информацию о городе Портланд: мэр этого города теперь беспартийный.

```
> db.towns.updateOne({name: 'Portland'}, {$unset: {'mayor.party': ''}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Посмотрим на обновленную запись

```
> db.towns.find({ name: 'Portland'}).pretty()
< {
    _id: ObjectId('6837cf3292b5677c8588eb19'),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams'
    }
  }
```

22.Задание 3.3.6:

Изменим информацию о самце единорога Pilot: теперь он любит и шоколад.

```
> db.unicorns.updateOne({name: 'Pilot', gender: 'm'}, {$addToSet: {loves: 'chocolate' }})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Посмотрим на обновленную запись

```
> db.unicorns.find({ name: 'Pilot'}).pretty()
< {
    _id: ObjectId('6837b4b192b5677c8588eb14'),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon',
      'chocolate'
    ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
```

23. Задание 3.3.7:

Изменим информацию о самке единорога Aurora: теперь она любит еще и сахар, и лимоны.

```
> db.unicorns.updateOne({name: 'Aurora', gender: 'f'}, {$addToSet: {loves: { $each: ['sugar', 'lemon']}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
learn>
```

Посмотрим на обновленную запись

```
> db.unicorns.find({ name: 'Aurora'}).pretty()
< {
    _id: ObjectId('6837b4b192b5677c8588eb0c'),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemon'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

24. Задание 3.4.1:

Создадим коллекцию towns, включающую заданные документы

```
> 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"
  }
  }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('6837e1ef92b5677c8588eb1b'),
      '1': ObjectId('6837e1ef92b5677c8588eb1c'),
      '2': ObjectId('6837e1ef92b5677c8588eb1d')
    }
  }
```

Удалим документы с беспартийными мэрами.

```
> db.towns.deleteMany({'mayor.party': {$exists: false}})
< {
    acknowledged: true,
    deletedCount: 3
  }
```

Проверим содержание коллекции.

```
{
  _id: ObjectId('6837e1ef92b5677c8588eb1c'),
  name: 'New York',
  popujatiuon: 22200000,
  last_sensus: 2009-07-31T00:00:00.000Z,
  famous_for: [
    'status of liberty',
    'food'
  ],
  mayor: {
    name: 'Michael Bloomberg',
    party: 'I'
  }
}
{
  _id: ObjectId('6837e1ef92b5677c8588eb1d'),
  name: 'Portland',
  popujatiuon: 528000,
  last_sensus: 2009-07-20T00:00:00.000Z,
  famous_for: [
    'beer',
    'food'
  ],
  mayor: {
    name: 'Sam Adams',
    party: 'D'
  }
}
```

Очистим коллекцию.

```
> db.towns.deleteMany({})
< {
    acknowledged: true,
    deletedCount: 3
  }
```

Просмотрим список доступных коллекций.

```
> show collections
< towns
  unicorns
learn >
```

25.Задание 4.1.1:

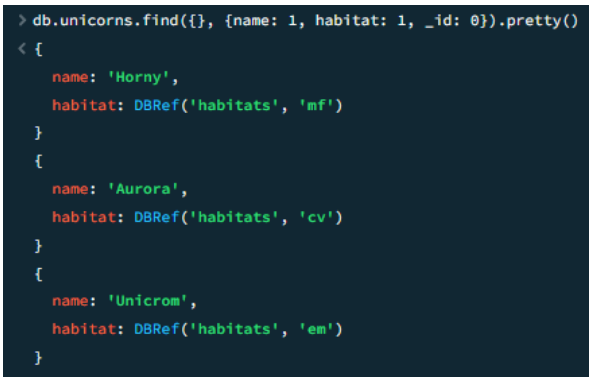
Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.

```
> db.habitats.insertMany([
    {
  _id: 'mf',
  fullName: 'Magic Forest',
  description: 'Magical forest with magical animals'
  },
  {
  _id: 'cv',
  fullName: 'Crystal Valley',
  description: 'Valley with crystals and a deep lake'
  },
  {
  _id: 'em',
  fullName: 'Sleeping mountains',
  description: 'Enchanted mountains with eternal snows'
  }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 'mf',
      '1': 'cv',
      '2': 'em'
    }
  }
```

Включим для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания

```
> db.unicorns.updateOne(
  {name: 'Horny'},
  {$set: {habitat: {$ref: 'habitats', $id: 'mf'}}}
  )
  db.unicorns.updateOne(
  {name: 'Aurora'},
  {$set: {habitat: {$ref: 'habitats', $id: 'cv'}}}
  )
  db.unicorns.updateOne(
  {name: 'Unicrom'},
  {$set: {habitat: {$ref: 'habitats', $id: 'em'}}}
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Проверим содержание коллекции единорогов.



```
> db.unicorns.find({}, {name: 1, habitat: 1, _id: 0}).pretty()
< {
    name: 'Horny',
    habitat: DBRef('habitats', 'mf')
  }
  {
    name: 'Aurora',
    habitat: DBRef('habitats', 'cv')
  }
  {
    name: 'Unicrom',
    habitat: DBRef('habitats', 'em')
  }
```

26.Задание 4.2.1:

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

```
> db.unicorns.createIndex({name: 1}, {unique: true})
< name_1
```

27.Задание 4.3.1:

Получите информацию о всех индексах коллекции unicorns .

```
learn > [ { v: 2, key: {_id: 1 }, name: '_id_'}
        { v: 2, key: { name: 1}, 'name_1', unique: true}
```

Удалите все индексы, кроме индекса для идентификатора.

```
> db.unicorns.dropIndex('name_1')
< { nIndexesWas: 2, ok: 1 }
```

Попытайтесь удалить индекс для идентификатора.

```
> db.unicorns.dropIndex('_id_')
✖ ▸ MongoServerError[InvalidOptions]: cannot drop _id index
```

28.Задание 4.4.1:

Создайте объемную коллекцию numbers, задействовав курсор

```
> for (i = 0; i < 100000; i++) {db.numbers.insertOne({value: i})}
< {
    acknowledged: true,
    insertedId: ObjectId('6837eb9c92b5677c858a71bd')
  }
```

Выберите последних четыре документа.

```
> db.numbers.find().sort({value: -1}).limit(4)
< {
    _id: ObjectId('6837eb9c92b5677c858a71bd'),
    value: 99999
  }
  {
    _id: ObjectId('6837eb9c92b5677c858a71bc'),
    value: 99998
  }
  {
    _id: ObjectId('6837eb9c92b5677c858a71bb'),
    value: 99997
  }
  {
    _id: ObjectId('6837eb9c92b5677c858a71ba'),
    value: 99996
  }
```

Проанализируйте план выполнения запроса 2. Сколько потребовалось времени на выполнение запроса?

```
> var explain = db.numbers.find().sort({value: -1}).limit(4).explain('executionStats')
  print('Time = ' + explain.executionStats.executionTimeMillis)
< Time = 70
```

Создайте индекс для ключа `value`.

```
> db.numbers.createIndex({value: 1})
< value_1
```

Получите информацию о всех индексах коллекции `numbres`.

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

Выполните запрос 2.

```
> db.numbers.find().sort({value: -1}).limit(4)
< {
    _id: ObjectId('6837eb9c92b5677c858a71bd'),
    value: 99999
  }
  {
    _id: ObjectId('6837eb9c92b5677c858a71bc'),
    value: 99998
  }
  {
    _id: ObjectId('6837eb9c92b5677c858a71bb'),
    value: 99997
  }
  {
    _id: ObjectId('6837eb9c92b5677c858a71ba'),
    value: 99996
  }
```

Проанализируйте план выполнения запроса с установленным индексом. Сколько потребовалось времени на выполнение запроса?

```
> var explainIndexed = db.numbers.find().sort({value: -1}).limit(4).explain('executionStats')
  print('Time = ' + explain.executionStats.executionTimeMillis)
< Time = 0
```

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

Без индекса запрос выполняется за 70 мс, а с индексом за 0 мс. Т.е. запрос с индексом более эффективен.