ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Мегафакультет трансляционных информационных технологий
Факультет инфокоммуникационных технологий

Дисциплина:

«Базы данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

«Работа с БД в СУБД MongoDB»

Выполнила:

студент группы K32421
Микулина Алиса Романовна

_____
(подпись)

Проверила:

Говорова Марина Михайловна

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург

2023 г.

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.

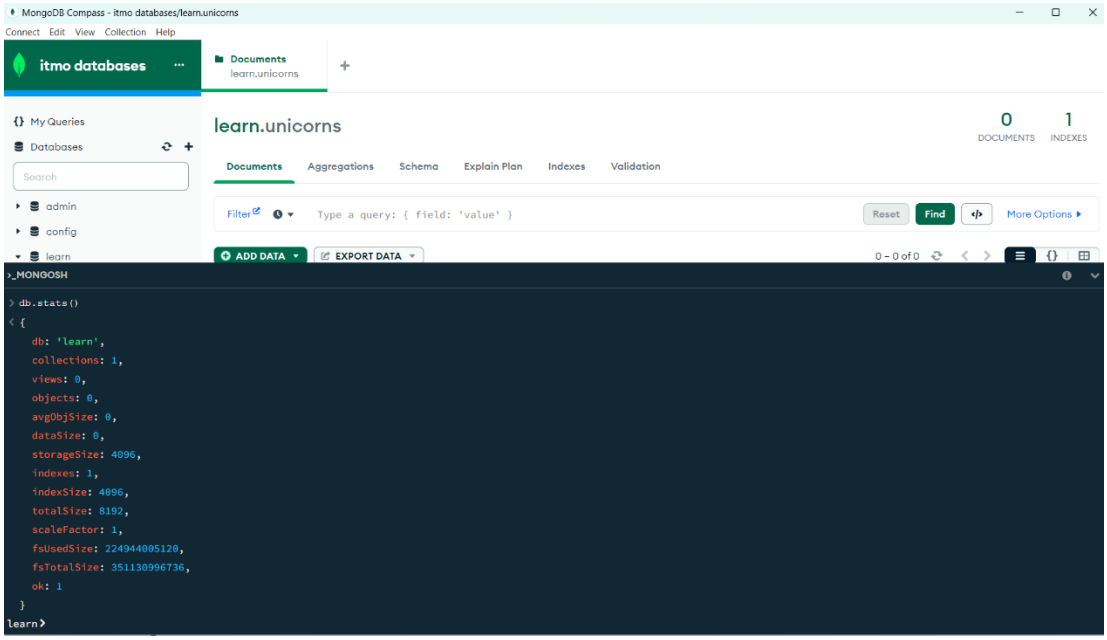
Оборудование: компьютерный класс.

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

Выполнение:

Практическое задание 8.1.1:

1. *Создайте базу данных learn.*



2. *Заполните коллекцию 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'});
```

```
>_MONGOSH

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("64733ebc722a0931f26acff0")
  }
}
learn>
```

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

```
{name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}
```

```
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.insert(document)
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64733f33722a0931f26acff1")
  }
}
```

4. *Проверьте содержимое коллекции с помощью метода find.*

```
db.unicorns.find()
{
  _id: ObjectId("64733ebc722a0931f26acfe6"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId("64733ebc722a0931f26acfe7"),
  name: 'Aurora',
  loves: [
    'carrot',
    'grape'
  ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
{
  _id: ObjectId("64733ebc722a0931f26acfe8"),
```

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.*

```
> db.unicorns.find({gender: 'm'}, {name: 1, _id: 0}).sort({name: 1}).toArray()
< [
    { name: 'Dunx' },
    { name: 'Horny' },
    { name: 'Kenny' },
    { name: 'Pilot' },
    { name: 'Raleigh' },
    { name: 'Roooooodles' },
    { name: 'Unicrom' }
  ]
learn>
```

```
> db.unicorns.find({gender: 'f'}, {name: 1, _id: 0}).sort({name: 1}).limit(3).toArray()
< [ { name: 'Aurora' }, { name: 'Ayna' }, { name: 'Leia' } ]
learn>
```

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

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}, {name: 1, _id: 0}).limit(1)
< {
    name: 'Aurora'
  }
> db.unicorns.findOne({gender: 'f', loves: 'carrot'}, {name: 1, _id: 0})
< {
    name: 'Aurora'
  }
learn>
```

Практическое задание 8.1.3:

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

```
> db.unicorns.find({gender: 'm'}, {_id: 0, gender: 0, loves: 0}).sort({name: 1}).toArray()
< [
    { name: 'Dunx', weight: 704, vampires: 165 },
    { name: 'Horny', weight: 600, vampires: 63 },
    { name: 'Kenny', weight: 690, vampires: 39 },
    { name: 'Pilot', weight: 650, vampires: 54 },
    { name: 'Raleigh', weight: 421, vampires: 2 },
    { name: 'Roooooodles', weight: 575, vampires: 99 },
    { name: 'Unicrom', weight: 984, vampires: 182 }
  ]
learn>
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find({}, {_id: 0, gender: 0, loves: 0}).sort({$natural: -1}).toArray()
< [
    { name: 'Dunx', weight: 704, vampires: 165 },
    { name: 'Nimue', weight: 540 },
    { name: 'Pilot', weight: 650, vampires: 54 },
    { name: 'Leia', weight: 601, vampires: 33 },
    { name: 'Raleigh', weight: 421, vampires: 2 },
    { name: 'Kenny', weight: 690, vampires: 39 },
    { name: 'Ayna', weight: 733, vampires: 40 },
    { name: 'Solnara', weight: 550, vampires: 80 },
    { name: 'Roooooodles', weight: 575, vampires: 99 },
    { name: 'Unicrom', weight: 984, vampires: 182 },
    { name: 'Aurora', weight: 450, vampires: 43 },
    { name: 'Horny', weight: 600, vampires: 63 }
  ]
learn>
```

Практическое задание 8.1.5:

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

```
> db.unicorns.find({}, {_id: 0, gender: 0, weight:0, vampires: 0, loves: {$slice : 1}}).toArray()
< [
    { name: 'Horny', loves: [ 'carrot' ] },
    { name: 'Aurora', loves: [ 'carrot' ] },
    { name: 'Unicrom', loves: [ 'energon' ] },
    { name: 'Roooooodles', loves: [ 'apple' ] },
    { name: 'Solnara', loves: [ 'apple' ] },
    { name: 'Ayna', loves: [ 'strawberry' ] },
    { name: 'Kenny', loves: [ 'grape' ] },
    { name: 'Raleigh', loves: [ 'apple' ] },
    { name: 'Leia', loves: [ 'apple' ] },
    { name: 'Pilot', loves: [ 'apple' ] },
    { name: 'Nimue', loves: [ 'grape' ] },
    { name: 'Dunx', loves: [ 'grape' ] }
  ]
learn>
```

Практическое задание 8.1.6:

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

```
> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 700}}, {_id: 0}).toArray()
< [
    {
      name: 'Solnara',
      loves: [ 'apple', 'carrot', 'chocolate' ],
      weight: 550,
      gender: 'f',
      vampires: 80
    },
    {
      name: 'Leia',
      loves: [ 'apple', 'watermelon' ],
      weight: 601,
      gender: 'f',
      vampires: 33
    },
    {
      name: 'Nimue',
      loves: [ 'grape', 'carrot' ],
      weight: 540,
      gender: 'f'
    }
  ]
learn>
```

Практическое задание 8.1.7:

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

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

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ `vampires`.

```
> db.unicorns.find({vampires: {$exists:false}}).toArray()
< [
    {
      _id: ObjectId("64733ebc722a0931f26acff0"),
      name: 'Nimue',
      loves: [ 'grape', 'carrot' ],
      weight: 540,
      gender: 'f'
    }
  ]
learn>
```

Практическое задание 8.1.9:

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

```
> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, loves: {$slice: 1}}).sort({name: 1}).toArray()
< [
    { name: 'Dunx', loves: [ 'grape' ] },
    { name: 'Horny', loves: [ 'carrot' ] },
    { name: 'Kenny', loves: [ 'grape' ] },
    { name: 'Pilot', loves: [ 'apple' ] },
    { name: 'Raleigh', loves: [ 'apple' ] },
    { name: 'Roooooodles', loves: [ 'apple' ] },
    { name: 'Unicrom', loves: [ 'energon' ] }
  ]
learn>
```

Практическое задание 8.2.1:

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

```
{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"}}
```

```
> db.towns.find()
< {
    _id: ObjectId("6473509b722a0931f26acff2"),
    name: 'Punxsutawney ',
    populatiuon: 6200,
    last_sensus: 2008-01-31T00:00:00.000Z,
    famous_for: [
      ''
    ],
    mayor: {
      name: 'Jim Wehrle'
    }
  }
  {
    _id: ObjectId("647350b3722a0931f26acff3"),
    name: 'New York',
    populatiuon: 22200000,
    last_sensus: 2009-07-31T00:00:00.000Z,
    famous_for: [
      'status of liberty',
      'food'
```

2. *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}).toArray()
< [
    {
      name: 'New York',
      mayor: { name: 'Michael Bloomberg', party: 'I' }
    }
  ]
learn>
```

3. *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}).toArray()
< [ { name: 'Punxsutawney ', mayor: { name: 'Jim Wehrle' } } ]
learn>
```

Практическое задание 8.2.2:

1. *Сформировать функцию для вывода списка самцов единорогов.*

```
> fn = function() {return this.gender=='m'}
< [Function: fn]
```

У MongoDB Compass какие-то приколы с функциями, они запускаются и бесконечно исполняются, крайне неудобно. Пришлось перезапускать приложение.

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

В теории вот это должно сработать, но это в теории, потому что функции в Compasse ломают все что только могут сломать

```
learn> var cursor = db.unicorns.find({"$where": fn}); null
```

```
learn> cursor.sort({name:1}).limit(2);null;
```

3. *Вывести результат, используя forEach.*

```
learn> cursor.forEach(function(obj){print(obj.name);})
```

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.find({gender: 'f', weight: {$gte: 500, $lte: 600}}, {_id: 0}).count(true)
< 2
learn>
```

Практическое задание 8.2.4:

Вывести список предпочтений.

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

Практическое задание 8.2.5:

Посчитать количество особей единорогов обоих полов.

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

Практическое задание 8.2.6:

1. *Выполнить команду:*

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

```
> db.unicorns.save({name: 'Barny', loves: ['grape'],
  weight: 340, gender: 'm'})
✖ ▸ TypeError: db.unicorns.save is not a function
> db.unicorns.countDocuments()
< 12
learn>
```

save не является функцией, так как устарел. В документации предлагают использовать insertOne вместо save.

2. *Проверить содержимое коллекции* `unicorns`.

```
> db.unicorns.insertOne({name: 'Barny', loves: ['grape'],
  weight: 340, gender: 'm'})
< {
    acknowledged: true,
    insertedId: ObjectId("647362db1cd3959898d489ca")
  }
> db.unicorns.countDocuments()
< 13
learn>
```

Практическое задание 8.2.7:

1. *Для самки единорога* `Ayna` *внести изменения в БД: теперь ее вес 800, она убила 51 вапмира.*
2. *Проверить содержимое коллекции* `unicorns`.

```
> db.unicorns.updateOne({name: 'Ayna'}, {$set: {weight: 800, vampires: 51}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name: 'Ayna'})
< {
    _id: ObjectId("64733ebc722a0931f26acfeb"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
learn>
```

Практическое задание 8.2.8:

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

Проверить содержимое коллекции `unicorns`.

```
> db.unicorns.updateOne({name: 'Raleigh'}, {$set: {loves: ['redbull']}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name: 'Raleigh'})
< {
    _id: ObjectId("64733ebc722a0931f26acfed"),
    name: 'Raleigh',
    loves: [
      'redbull'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
learn>
```

Практическое задание 8.2.9:

1. *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*

```
> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, vampires: 1}).toArray()
< [
    { name: 'Horny', vampires: 63 },
    { name: 'Unicrom', vampires: 182 },
    { name: 'Roooooodles', vampires: 99 },
    { name: 'Kenny', vampires: 39 },
    { name: 'Raleigh', vampires: 2 },
    { name: 'Pilot', vampires: 54 },
    { name: 'Dunx', vampires: 165 },
    { name: 'Barny', vampires: 0 }
  ]
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 8,
    modifiedCount: 8,
    upsertedCount: 0
  }
```

2. *Проверить содержимое коллекции* `unicorns`.

```
> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, vampires: 1}).toArray()
< [
    { 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: 'Dunx', vampires: 170 },
    { name: 'Barny', vampires: 5 }
  ]
learn>
```

Практическое задание 8.2.10:

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

```
> db.towns.find({name: 'Portland'})
< {
    _id: ObjectId("647350c6722a0931f26acff4"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams',
      party: 'D'
    }
  }
> db.towns.updateOne({name: 'Portland'}, {$unset: {'mayor.party': 1}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

2. *Проверить содержимое коллекции* `towns`.

```
> db.towns.find({name: 'Portland'})
< {
    _id: ObjectId("647350c6722a0931f26acff4"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams'
    }
  }
learn>
```

Практическое задание 8.2.11:

1. *Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.*
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("64733ebc722a0931f26acfef"),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon',
      'chocolate'
    ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
learn>
```

Практическое задание 8.2.12:

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

```
> db.unicorns.updateOne({name: 'Aurora'}, {$addToSet: {loves: {$each: ['sugar', 'lemons']}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name: 'Aurora'})
< {
    _id: ObjectId("64733ebc722a0931f26acfe7"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemons'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
learn>
```

Практическое задание 8.2.13:

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"}}
```

Коллекция уже была создана ранее. В ней два беспартийных мера.

2. *Удалите документы с беспартийными мэрами.*
3. *Проверьте содержание коллекции.*

```
> db.towns.countDocuments()
< 3
> db.towns.deleteMany({'mayor.party': {$exists: false}})
< {
    acknowledged: true,
    deletedCount: 2
  }
> db.towns.countDocuments()
< 1
```

4. *Очистите коллекцию.*

```
> db.towns.remove({})
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.towns.countDocuments()
< 0
```

5. *Просмотрите список доступных коллекций.*

```
> db.towns.drop()
< true
> show collections
< unicorns
```

Практическое задание 8.3.1:

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

```
> db.areas.insertOne({_id: 1, name: 'field', description: 'a field covered with soft green grass'})
< {
    acknowledged: true,
    insertedId: 1
  }
> db.areas.insertOne({_id: 2, name: 'cloud', description: 'a soft fluffy cloud 3 km above the field'})
< {
    acknowledged: true,
    insertedId: 2
  }
> db.areas.find()
< {
    _id: 1,
    name: 'field',
    description: 'a field covered with soft green grass'
  }
  {
    _id: 2,
    name: 'cloud',
    description: 'a soft fluffy cloud 3 km above the field'
  }
learn>
```

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

```
> db.unicorns.updateMany({name: {$in: ['Unicrom', 'Raleigh', 'Pilot']}}, {$set: {area: {$ref: 'areas', $id: 1}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 3,
    modifiedCount: 3,
    upsertedCount: 0
  }
> db.unicorns.updateMany({name: {$in: ['Dunx', 'Nimue']}}, {$set: {area: {$ref: 'areas', $id: 2}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
learn>
```

3. *Проверьте содержание коллекции единорогов.*

```
db.unicorns.find({}, {_id:0, name: 1, area: 1}).toArray()
[
  { name: 'Horny' },
  { name: 'Aurora' },
  { name: 'Unicrom', area: DBRef("areas", 1) },
  { name: 'Roooooodles' },
  { name: 'Solnara' },
  { name: 'Ayna' },
  { name: 'Kenny' },
  { name: 'Raleigh', area: DBRef("areas", 1) },
  { name: 'Leia' },
  { name: 'Pilot', area: DBRef("areas", 1) },
  { name: 'Nimue', area: DBRef("areas", 2) },
  { name: 'Dunx', area: DBRef("areas", 2) },
  { name: 'Barny' }
]
earn>
```

Практическое задание 8.3.2:

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: f17e583e-c7f1-49c9-841c-b03343ef9663: Collection learn.unicorns ( 05063741-c136-40d1-b609-8e161416562c ) :: caused by :: E11000 duplicat
learn>
```

Практическое задание 8.3.3:

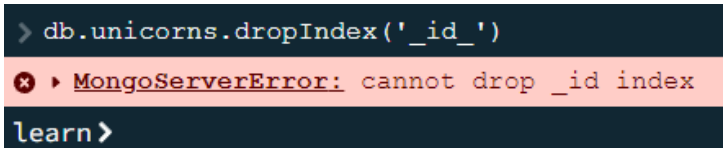
1. *Получите информацию о всех индексах коллекции* `unicorns`.

```
> db.unicorns.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
  ]
learn>
```

2. *Удалите все индексы, кроме индекса для идентификатора.*

```
> db.unicorns.dropIndex('name_1')
< { nIndexesWas: 2, ok: 1 }
> db.unicorns.getIndexes()
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
learn>
```

3. *Попытайтесь удалить индекс для идентификатора.*



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`*)*

4. *Создайте индекс для ключа* `value`*.*
5. *Получите информацию о всех индексах коллекции* `numbres`*.*
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?*

Без индекса запрос db.numbers.find().sort({ value: 1 }).limit(4) выполнялся 4 миллисекунды, с индексом – одну.

Вывод:

В данной работе я овладела 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.