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

Отчет

по лабораторной работе «Работа с БД в СУБД MongoDB»

по дисциплине «**Базы данных**»

Автор: Скороходова Елена

Факультет: ИКТ

Группа: K32392

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

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.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД MongoDB 4 6.0.6

Практическое задание и выполнение:

Практическое задание 8.1.1:

1) Создайте базу данных learn.

```
> use learn
< switched to db learn
> db.createCollection('unicorns')
MongoServerError: Collection learn.unicorns already exists.
> db.unicorns.find().sort()
<
> db.unicorns.find().count()
< 0
> 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});
```

2) Заполните коллекцию единорогов unicorns:

```
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("6474ae951e6f619ad71aed5c")
  }
}
db.unicorns.find().count()
11
db.unicorns.find()
{
  _id: ObjectId("6474ae931e6f619ad71aed52"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
```

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

```
document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires:
165}
);
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
```

4) Проверьте содержимое коллекции с помощью метода find.

```
{
  _id: ObjectId("6474ae951e6f619ad71aed5c"),
  name: 'Nimue',
  loves: [
    'grape',
    'carrot'
  ],
  weight: 540,
  gender: 'f'
}
{
  _id: ObjectId("6474b1de1e6f619ad71aed5e"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
```

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'}).sort({name: 1});
{
  _id: ObjectId("6474b1de1e6f619ad71aed5e"),
  name: 'Dunx',
  loves: [
    'grape',
    'watermelon'
  ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
{
  _id: ObjectId("6474ae931e6f619ad71aed52"),
  name: 'Horny',
  loves: [
    'carrot',
    'papaya'
  ],
  weight: 600,
  gender: 'm',
  vampires: 63
}
{
  _id: ObjectId("6474ae941e6f619ad71aed58"),
  name: 'Kenny'
```

```
> db.unicorns.find({gender: 'f'}).sort({name: 1}).limit(3);
< {
    _id: ObjectId("6474ae931e6f619ad71aed53"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
  {
    _id: ObjectId("6474ae941e6f619ad71aed57"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
  {
    _id: ObjectId("6474ae941e6f619ad71aed5a"),
```

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'}, {_id: 0, name: 1}).sort({name: 1}).limit(1);
< {
    name: 'Aurora'
  }
```

```
> db.unicorns.findOne({gender: 'f', loves: 'carrot'});
< {
    _id: ObjectId("6474ae931e6f619ad71aed53"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

```
> db.unicorns.find({gender: 'f', loves: 'carrot'}, {_id: 0, name: 1}).sort({name: 1});
< {
    name: 'Aurora'
  }
  {
    name: 'Nimue'
  }
  {
    name: 'Solnara'
  }
```

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'}, {loves: 0, gender: 0});
< {
    _id: ObjectId("6474ae931e6f619ad71aed52"),
    name: 'Horny',
    weight: 600,
    vampires: 63
  }
  {
    _id: ObjectId("6474ae931e6f619ad71aed54"),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  }
  {
    _id: ObjectId("6474ae931e6f619ad71aed55"),
    name: 'Roooooodles',
    weight: 575,
    vampires: 99
  }
  {
    _id: ObjectId("6474ae941e6f619ad71aed58"),
    name: 'Kenny',
    weight: 690,
    vampires: 39
  }
  {
```

Практическое задание 8.1.4:

Вывести список единорогов в обратном порядке добавления.

```
> db.unicorns.find({}).sort({$natural: -1});
< {
    _id: ObjectId("6474b1de1e6f619ad71aed5e"),
    name: 'Dunx',
    loves: [
      'grape',
      'watermelon'
    ],
    weight: 704,
    gender: 'm',
    vampires: 165
  }
  {
    _id: ObjectId("6474ae951e6f619ad71aed5c"),
    name: 'Nimue',
    loves: [
      'grape',
      'carrot'
    ],
    weight: 540,
```

Практическое задание 8.1.5:

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

Практическое задание 8.1.6:

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

```
> db.unicorns.find({gender: 'f', weight: {$gt: 500, $lt:700}}, {_id: 0, });
< {
    name: 'Solnara',
    loves: [
      'apple',
      'carrot',
      'chocolate'
    ],
    weight: 550,
    gender: 'f',
    vampires: 80
  }
  {
    name: 'Leia',
    loves: [
      'apple',
      'watermelon'
    ],
    weight: 601,
    gender: 'f',
```

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: {$gt: 500}, loves: { $all: ['grape', 'lemon']}}, {_id: 0});
< {
    name: 'Kenny',
    loves: [
      'grape',
      'lemon'
    ],
    weight: 690,
    gender: 'm',
    vampires: 39
  }
```

Практическое задание 8.1.8:

Найти всех единорогов, не имеющих ключ vampires.

```
> db.unicorns.find({gender: 'f', vampires:{$exists: false}}, {_id: 0, name: 1});
< {
    name: 'Nimue'
  }
```

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});
< {
    name: 'Dunx',
    loves: [
      'grape'
    ]
  }
  {
    name: 'Horny',
    loves: [
      'carrot'
    ]
  }
  {
    name: 'Kenny',
    loves: [
      'grape'
    ]
```

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.insert({name: "Punxsutawney ",
  populatiuon: 6200,
  last_sensus: ISODate("2008-01-31"),
  famous_for: [""],
  mayor: {
  name: "Jim Wehrle"
  }});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("64772d81ba4f00dc48d2f565")
    }
  }
> db.towns.insert({name: "New York",
  populatiuon: 22200000,
  last_sensus: ISODate("2009-07-31"),
  famous_for: ["status of liberty", "food"],
  mayor: {
```

```
> db.towns.find().count();
< 3
```

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}).sort({name: 1});
< {
    name: 'New York',
    mayor: {
      name: 'Michael Bloomberg',
      party: 'I'
    }
  }
```

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}).sort({name: 1});
< {
    name: 'Punxsutawney ',
    mayor: {
      name: 'Jim Wehrle'
    }
  }
```

Практическое задание 8.2.2:

3) Сформировать функцию для вывода списка самцов единорогов.

```
> un_male = function() {return this.gender == 'm';}
< [Function: un_male]
> db.unicorns.find(un_male);
  db.unicorns.find(un_male);
```

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

```
> males = function() {return this.gender == 'm';}
< [Function: males]
> var cursor = db.unicorns.find({'$where': males}).sort({name: 1}).limit(2);null;
  var cursor = db.unicorns.find({'$where': males}).sort({name: 1}).limit(2);null;
```

5) Вывести результат, используя forEach.

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

Практическое задание 8.2.3:

Вывести количество самок единорогов весом от полутонны до 600 кг.

```
> db.unicorns.find({gender: 'f', weight: {$gt: 500, $lt: 600}}).count();
< 2
```

Практическое задание 8.2.4:

Вывести список предпочтений.

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

Практическое задание 8.2.5:

Подсчитать количество особей единорогов обоих полов.

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

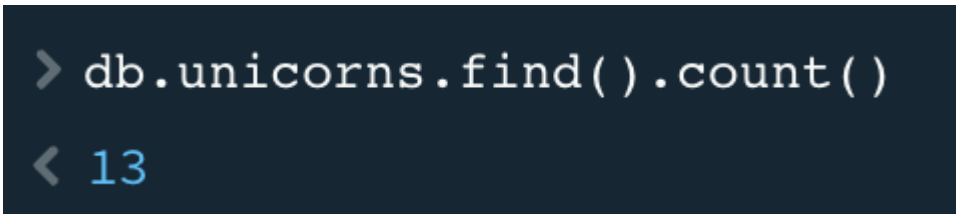
Практическое задание 8.2.6:

1. Выполнить команду:
db.unicorns.save({name: 'Barny', loves: ['grape'],
weight: 340, gender: 'm'})
2. Проверить содержимое коллекции unicorns.

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

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

< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("647769356dcec0e76616b928")
    }
  }
```



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("6474ae941e6f619ad71aed57"),
    name: 'Ayna',
    loves: [
      'strawberry',
      'lemon'
    ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
```

Практическое задание 8.2.8:

1. Для самца единорога Raleigh внести изменения в БД: теперь он любит рэдбул.
2. 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("6474ae941e6f619ad71aed59"),
    name: 'Raleigh',
    loves: [
      'redbull'
    ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
```

Практическое задание 8.2.9:

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

```
> db.unicorns.find({gender: 'm'})
< {
    _id: ObjectId("6474ae931e6f619ad71aed52"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 63
  }
  {
    _id: ObjectId("6474ae931e6f619ad71aed54"),
    name: 'Unicrom',
    loves: [
      'energon',
      'redbull'
    ],
    weight: 984,
    gender: 'm',
    vampires: 182
  }
  {
    _id: ObjectId("6474ae931e6f619ad71aed55"),
```

```
> db.unicorns.updateMany({gender: 'm'}, {$inc: {vampires: 5}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 8,
    modifiedCount: 8,
    upsertedCount: 0
  }
> db.unicorns.find({gender: 'm'}, {_id: 0, name: 1, vampires: 1}).sort({name:1})
< {
    name: 'Barny',
    vampires: 5
  }
  {
    name: 'Dunx',
    vampires: 170
  }
  {
    name: 'Horny',
    vampires: 68
  }
  {
    name: 'Kenny',
    vampires: 44
  }
```

Практическое задание 8.2.10:

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

```
{
  _id: ObjectId("64772dc7ba4f00dc48d2f567"),
  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
  }
> db.towns.find({name: 'Portland'})
< {
    _id: ObjectId("64772dc7ba4f00dc48d2f567"),
    name: 'Portland',
    populatiuon: 528000,
    last_sensus: 2009-07-20T00:00:00.000Z,
    famous_for: [
      'beer',
      'food'
    ],
    mayor: {
      name: 'Sam Adams'
    }
```

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("6474ae941e6f619ad71aed5b"),
    name: 'Pilot',
    loves: [
      'apple',
      'watermelon',
      'chocolate'
    ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
```

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', 'lemon']}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name: 'Aurora'})
< {
    _id: ObjectId("6474ae931e6f619ad71aed53"),
    name: 'Aurora',
    loves: [
      'carrot',
      'grape',
      'sugar',
      'lemon'
    ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
```

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"}}

2) Удалите документы с беспартийными мэрами.
3) Проверьте содержание коллекции.
4) Очистите коллекцию.
5) Просмотрите список доступных коллекций.

```
> db.towns.find().count()
< 3
> db.towns.deleteMany({'mayor.party': {$exists: false}})
< {
    acknowledged: true,
    deletedCount: 2
  }
> db.towns.find().count()
< 1
> db.towns.drop()
< true
> show collections
< unicorns
  users
```

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.createCollection('areas')
< { ok: 1 }
> db.areas.insertMany([{'_id': 'Rainbow', 'name': 'Pretty rainbow', 'description': 'Soft, candy and friendly'}, {'_id': 'Castle', 'name': 'Dark castle', 'description': 'Far from ev
< {
    acknowledged: true,
    insertedIds: {
      '0': 'Rainbow',
      '1': 'Castle'
    }
  }
> db.areas.find()
< {
    _id: 'Rainbow',
    name: 'Pretty rainbow',
    description: 'Soft, candy and friendly'
  }
  {
    _id: 'Castle',
    name: 'Dark castle',
    description: 'Far from everything'
  }
```

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

```
> db.unicorns.updateOne({name: 'Horny'}, {$set: {area: {$ref: 'areas', $id: 'Rainbow'}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
> db.unicorns.find({name: 'Horny'})
< {
    _id: ObjectId("6474ae931e6f619ad71aed52"),
    name: 'Horny',
    loves: [
      'carrot',
      'papaya'
    ],
    weight: 600,
    gender: 'm',
    vampires: 68,
    area: DBRef("areas", 'Rainbow')
  }
```

3)

4) Проверьте содержание коллекции единорогов

```
> db.unicorns.find({area: {$exists: true}}, {_id: 0, name: 1, area: 1})
< {
    name: 'Horny',
    area: DBRef("areas", 'Rainbow')
  }
  {
    name: 'Pilot',
    area: DBRef("areas", 'Castle')
  }
```

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' ]
```

Практическое задание 8.3.3:

1) Получите информацию о всех индексах коллекции unicorns .
2)

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

3) Удалите все индексы, кроме индекса для идентификатора.
4) Попытайтесь удалить индекс для идентификатора.

```
> db.unicorns.dropIndex('name_1')
< {
    nIndexesWas: 2,
    ok: 1,
    '$clusterTime': {
      clusterTime: Timestamp({ t: 1685562395, i: 7 }),
      signature: {
        hash: Binary(Buffer.from("48cd71a5587dd63f70eeb4f8e5ad5e965394ef2f", "hex"), 0),
        keyId: Long("7176326465359708162")
      }
    },
    operationTime: Timestamp({ t: 1685562395, i: 7 })
  }
> db.unicorns.getIndexes();
< [ { v: 2, key: { _id: 1 }, name: '_id_' } ]
> 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
```

Практическое задание 8.3.4:

1) Создайте объемную коллекцию numbers, задействовав курсор:

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

```
> db.createCollection('numbers')
< { ok: 1 }
> for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}
  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?

```
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
  db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
```

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

```
> db.numbers.find().sort({_id: -1}).limit(4)
  db.numbers.find().sort({_id: -1}).limit(4)
```

```
> db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
  db.numbers.explain("executionStats").find().sort({$natural: -1}).limit(4)
```

С индексом запрос работает быстрее.

Выводы:

В процессе работы были получены навыки работы с CRUD-операциями, с вложенными объектами в коллекции базы данных MongoDB, агрегации и изменения данных, со ссылками и индексами в базе данных MongoDB. Так же работа с MongoDB может быть выполнена и используя облачный сервис MongoDB Atlas, а для наибольшего удобства – и с использованием графического интерфейса MongoDB Compass.