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

# ОТЧЕТ

## ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

## «Работа с БД в СУБД MongoDB»

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

**Обучающийся** Люсин Дмитрий Витальевич
**Факультет** прикладной информатики
**Группа** K3239
**Направление подготовки** 09.03.03 Прикладная информатика
**Образовательная программа** Мобильные и сетевые технологии 2023
**Преподаватель** Говорова Марина Михайловна

Санкт-Петербург
2025/2026

**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

**Практическая часть:** Практические задания 2.1.1 – 4.1.1

**Практическое задание 2.1.1:**

Заполнение коллекции unicorns и вставка в коллекцию документа:

```
test> use learn
switched to db learn
learn> db.unicorns.insertMany([
...   {name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63},
...   {name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43},
...   {name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182},
...   {name: 'Roooooodles', loves: ['apple'], weight: 575, gender: 'm', vampires: 99},
...   {name: 'Solnara', loves: ['apple', 'carrot', 'chocolate'], weight: 550, gender: 'f', vampires: 80},
...   {name: 'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40},
...   {name: 'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39},
...   {name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2},
...   {name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33},
...   {name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54},
...   {name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683751f4ffe7cdce97d861e1'),
    '1': ObjectId('683751f4ffe7cdce97d861e2'),
    '2': ObjectId('683751f4ffe7cdce97d861e3'),
    '3': ObjectId('683751f4ffe7cdce97d861e4'),
    '4': ObjectId('683751f4ffe7cdce97d861e5'),
    '5': ObjectId('683751f4ffe7cdce97d861e6'),
    '6': ObjectId('683751f4ffe7cdce97d861e7'),
    '7': ObjectId('683751f4ffe7cdce97d861e8'),
    '8': ObjectId('683751f4ffe7cdce97d861e9'),
    '9': ObjectId('683751f4ffe7cdce97d861ea'),
    '10': ObjectId('683751f4ffe7cdce97d861eb')
learn>
}
learn> document = ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165});
{
  name: 'Dunx',
  loves: [ 'grape', 'watermelon' ],
  weight: 704,
  gender: 'm',
  vampires: 165
}
learn> db.unicorns.insert(document);
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68375259ffe7cdce97d861ec') }
}
```

Проверка содержимого:

```
learn> db.unicorns.find()
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e1'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e3'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
```

**Практическое задание 2.2.1:**

Вывод списка самцов, первых 3-ёх самок и сортировка списка:

```
learn> db.unicorns.find({gender: "m"})
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e1'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e3'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e4'),
    name: 'Roooooodles',
    loves: [ 'apple' ],
    weight: 575,
    gender: 'm',
    vampires: 99
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e7'),
```

```
learn> db.unicorns.find({gender: "f"}).limit(3)
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e5'),
    name: 'Solnara',
    loves: [ 'apple', 'carrot', 'chocolate' ],
    weight: 550,
    gender: 'f',
    vampires: 80
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e6'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  }
]
```

```
learn> db.unicorns.find({gender: "f"}).sort({name: 1})
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e6'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 733,
    gender: 'f',
    vampires: 40
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e9'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861eb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
```

Самки, любящие carrot:

```
learn> db.unicorns.findOne({gender: "f", loves: "carrot"})
{
  _id: ObjectId('683751f4ffe7cdce97d861e2'),
  name: 'Aurora',
  loves: [ 'carrot', 'grape' ],
  weight: 450,
  gender: 'f',
  vampires: 43
}
learn> db.unicorns.find({gender: "f", loves: "carrot"}).limit(1)
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e2'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

**Практическое задание 2.2.2:**

Список самцов без предпочтений и поля:

```
learn> db.unicorns.find({gender: 'm'}, {loves: 0, gender: 0})
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861e1'),
    name: 'Horny',
    weight: 600,
    vampires: 63
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e3'),
    name: 'Unicrom',
    weight: 984,
    vampires: 182
  },
```

**Практическое задание 2.2.3:**

Вывод списка в обратном порядке добавления:

```
learn> db.unicorns.find().sort({$natural: -1})
[
  {
    _id: ObjectId('68375259ffe7cdce97d861ec'),
    name: 'Dunx',
    loves: [ 'grape', 'watermelon' ],
    weight: 704,
    gender: 'm',
    vampires: 165
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861eb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  },
```

**Практическое задание 2.1.4:**

Список с названием первого любимого предпочтения, исключая идентификатор:

```
learn> db.unicorns.find({}, {loves: {$slice: 1}, _id: 0})
[
  {
    name: 'Horny',
    loves: [ 'carrot' ],
    weight: 600,
    gender: 'm',
    vampires: 63
  },
  {
    name: 'Aurora',
    loves: [ 'carrot' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  },
```

**Практическое задание 2.3.1:**

Список самок с весом от 500кг до 700кг, исключая идентификатор:

```
learn> db.unicorns.find({gender: "f", weight: {$gte: 500, $lte: 700}}, {_id: 0})
[
  {
    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'
  }
]
```

**Практическое задание 2.3.2:**

список самцов весом от полутонны и предпочитающих grape и lemon, исключая вывод идентификатора:

```
learn> 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
  }
]
```

**Практическое задание 2.3.3:**

Всех единороги, не имеющих ключ vampires:

```
learn> db.unicorns.find({vampires: {$exists: false}})
[
  {
    _id: ObjectId('683751f4ffe7cdce97d861eb'),
    name: 'Nimue',
    loves: [ 'grape', 'carrot' ],
    weight: 540,
    gender: 'f'
  }
]
```

**Практическое задание 2.3.4:**

Упорядоченный список имен самцов с информацией об их первом предпочтении:

```
learn> db.unicorns.find({gender: "m"}, {name: 1, loves: {$slice: 1}}).sort({name: 1})
[
  {
    _id: ObjectId('68375259ffe7cdce97d861ec'),
    name: 'Dunx',
    loves: [ 'grape' ]
  },
  {
    _id: ObjectId('683751f4ffe7cdce97d861e1'),
    name: 'Horny',
    loves: [ 'carrot' ]
  },
```

## Практическое задание 3.1.1:

Создание коллекции и формирование запросов:

```
learn> db.towns.insertMany([
...   {
...     name: "Punxsutawney",
...     populatiuon: 6200,
...     last_sensus: ISODate("2008-01-31"),
...     famous_for: [""],
...     mayor: { name: "Jim Wehrle" }
...   },
...   {
...     name: "New York",
...     populatiuon: 22200000,
...     last_sensus: ISODate("2009-07-31"),
...     famous_for: ["status of liberty", "food"],
...     mayor: { name: "Michael Bloomberg", party: "I" }
...   },
...   {
...     name: "Portland",
...     populatiuon: 528000,
...     last_sensus: ISODate("2009-07-20"),
...     famous_for: ["beer", "food"],
...     mayor: { name: "Sam Adams", party: "D" }
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('683781b463b9c722c1d861e0'),
    '1': ObjectId('683781b463b9c722c1d861e1'),
    '2': ObjectId('683781b463b9c722c1d861e2')
  }
}
```

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

## Практическое задание 3.1.2:

Формирование функции, создание курсора и вывод результата:

```
learn> function listMaleUnicorn() {
...   const cursor = db.unicorns.find({gender: "m"}).sort({name: 1}).limit(2);
...   cursor.forEach(function(obj) {
...     print(obj.name);
...   });
... }
[Function: listMaleUnicorn]
learn> listMaleUnicorn()
Dunx
Horny
```

## Практическое задание 3.2.1:

Подсчёт самок с нужным весом:

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

**Практическое задание 3.2.2:**

Вывод списка предпочтений:

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

**Практическое задание 3.2.3:**

Кол-во особей обоих полов:

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

**Практическое задание 3.3.1:**

Выполняем команду save и получаем ошибку. Вместо save используем insertOne:

```
learn> db.unicorns.save({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
TypeError: db.unicorns.save is not a function
learn> db.unicorns.insertOne({name: 'Barny', loves: ['grape'], weight: 340, gender: 'm'})
{
  acknowledged: true,
  insertedId: ObjectId('683789b963b9c722c1d861ef')
}
```

Проверяем:

```
{
  _id: ObjectId('683789b963b9c722c1d861ef'),
  name: 'Barny',
  loves: [ 'grape' ],
  weight: 340,
  gender: 'm'
}
```

**Практическое задание 3.3.2:**

Вносим изменения и проверяем:

```
learn> db.unicorns.updateOne(
...   { name: "Ayna"},
...   { $set: { weight: 800, vampires: 51 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Ayna" })
[
  {
    _id: ObjectId('683784c763b9c722c1d861e8'),
    name: 'Ayna',
    loves: [ 'strawberry', 'lemon' ],
    weight: 800,
    gender: 'f',
    vampires: 51
  }
]
```

## Практическое задание 3.3.3:

Добавляем данные в «любимое» и проверяем:

```
learn> db.unicorns.updateOne(
...   { name: "Raleigh",},
...   { $push: { loves: "redbull" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Raleigh" })
[
  {
    _id: ObjectId('683784c763b9c722c1d861ea'),
    name: 'Raleigh',
    loves: [ 'apple', 'sugar', 'redbull' ],
    weight: 421,
    gender: 'm',
    vampires: 2
  }
]
```

## Практическое задание 3.3.4:

Увеличиваем кол-во вампиров и сравниваем было/стало:

```
learn> db.unicorns.updateMany(
...   { gender: "m" },
...   { $inc: { vampires: 5 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 8,
  modifiedCount: 8,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId('683784c763b9c722c1d861e3'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 63
},
```

```
{
  _id: ObjectId('683784c763b9c722c1d861e3'),
  name: 'Horny',
  loves: [ 'carrot', 'papaya' ],
  weight: 600,
  gender: 'm',
  vampires: 68
},
```

## Практическое задание 3.3.5:

Вносим изменения и проверяем:

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

```
{
  _id: ObjectId('683781b463b9c722c1d861e2'),
  name: 'Portland',
  populatiuon: 528000,
  last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
  famous_for: [ 'beer', 'food' ],
  mayor: { name: 'Sam Adams' }
}
```

## Практическое задание 3.3.6:

Снова вносим изменения и проверяем:

```
learn> db.unicorns.updateOne(
...   { name: "Pilot"},
...   { $addToSet: { loves: "chocolate" } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Pilot" })
[
  {
    _id: ObjectId('683784c763b9c722c1d861ec'),
    name: 'Pilot',
    loves: [ 'apple', 'watermelon', 'chocolate' ],
    weight: 650,
    gender: 'm',
    vampires: 59
  }
]
```

## Практическое задание 3.3.7:

Вносим изменения и проверяем:

```
learn> db.unicorns.updateOne(
...   { name: "Aurora"},
...   { $addToSet: { loves: { $each: ["sugar", "lemon"] } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn> db.unicorns.find({ name: "Aurora" })
[
  {
    _id: ObjectId('683784c763b9c722c1d861e4'),
    name: 'Aurora',
    loves: [ 'carrot', 'grape', 'sugar', 'lemon' ],
    weight: 450,
    gender: 'f',
    vampires: 43
  }
]
```

## Практическое задание 3.4.1:

Производим удаление доков с мэрами без партии:

```
..     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('68378ff863b9c722c1d861f0'),
    '1': ObjectId('68378ff863b9c722c1d861f1'),
    '2': ObjectId('68378ff863b9c722c1d861f2')
  }
}
learn> db.towns.deleteMany({ "mayor.party": { $exists: false } })
{ acknowledged: true, deletedCount: 1 }
learn> db.towns.find()
[
  {
    _id: ObjectId('68378ff863b9c722c1d861f1'),
    name: 'New York',
    popujatiuon: 22200000,
    last_sensus: ISODate('2009-07-31T00:00:00.000Z'),
    famous_for: [ 'status of liberty', 'food' ],
    mayor: { name: 'Michael Bloomberg', party: 'I' }
  },
  {
    _id: ObjectId('68378ff863b9c722c1d861f2'),
    name: 'Portland',
    popujatiuon: 528000,
    last_sensus: ISODate('2009-07-20T00:00:00.000Z'),
    famous_for: [ 'beer', 'food' ],
    mayor: { name: 'Sam Adams', party: 'D' }
```

Вычищаем всю коллекцию и проверяем:

```
learn> db.towns.deleteMany({})
{ acknowledged: true, deletedCount: 2 }
learn> show collections
towns
unicorns
```

## Практическое задание 4.1.1:

Создаём коллекцию зон обитания, включаем ссылки на зоны обитания и проверяем:

```
learn> db.habitats.insertMany([
...   {
...     _id: "mt",
...     name: "Mountain Region",
...     description: "High-altitude rocky terrain with cold climate"
...   },
...   {
...     _id: "frst",
...     name: "Forest Zone",
...     description: "Lush forest area with dense vegetation and rivers"
...   },
...   {
...     _id: "pln",
...     name: "Plains",
...     description: "Open flatlands with grass and scattered trees"
...   }
... ])
{
  acknowledged: true,
  insertedIds: { '0': 'mt', '1': 'frst', '2': 'pln' }
}
```

```
learn> db.unicorns.find({ name: { $in: ["Horny", "Unicrom", "Leia"]}})
[
  {
    _id: ObjectId('683792bf63b9c722c1d861f3'),
    name: 'Horny',
    loves: [ 'carrot', 'papaya' ],
    weight: 600,
    gender: 'm',
    vampires: 63,
    habitat: DBRef('habitats', 'pln')
  },
  {
    _id: ObjectId('683792bf63b9c722c1d861f5'),
    name: 'Unicrom',
    loves: [ 'energon', 'redbull' ],
    weight: 984,
    gender: 'm',
    vampires: 182,
    habitat: DBRef('habitats', 'mt')
  },
  {
    _id: ObjectId('683792bf63b9c722c1d861fb'),
    name: 'Leia',
    loves: [ 'apple', 'watermelon' ],
    weight: 601,
    gender: 'f',
    vampires: 33,
    habitat: DBRef('habitats', 'frst')
  }
]
```

```
learn> db.unicorns.updateOne(
...   { name: "Horny" },
...   { $set: { habitat: { $ref: "habitats", $id: "pln" } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

learn> db.unicorns.updateOne(
...   { name: "Unicrom" },
...   { $set: { habitat: { $ref: "habitats", $id: "mt" } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
learn>

learn> db.unicorns.updateOne(
...   { name: "Leia" },
...   { $set: { habitat: { $ref: "habitats", $id: "frst" } } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

## Практическое задание 4.2.1:

Проверяем:

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

**Практическое задание 4.3.1:**

Получаем информацию об индексах, удаляем всё, кроме индекса идентификатора и пробуем удалить и его:

```
learn> db.unicorns.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { name: 1 }, name: 'name_1', unique: true }
]
learn> db.unicorns.dropIndexes()
{
  nIndexesWas: 2,
  msg: 'non-_id indexes dropped for collection',
  ok: 1
}
learn> db.unicorns.dropIndex("_id_")
MongoServerError[InvalidOptions]: cannot drop _id index
```

**Практическое задание 4.4.1:**

Создаём объёмную коллекцию:

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

{
  acknowledged: true,
  insertedId: ObjectId('68379862171a15f7e7d9e87f')
}
learn>
```

Время выполнения на 4 документа:

```
learn> db.numbers.find().sort({ value: -1 }).limit(4).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'learn.numbers',
    parsedQuery: {},
    indexFilterSet: false,
    queryHash: 'BA27D965',
    planCacheShapeHash: 'BA27D965',
    planCacheKey: '7A892B81',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'SORT',
      sortPattern: { value: -1 },
      memLimit: 104857600,
      limitAmount: 4,
      type: 'simple',
      inputStage: { stage: 'COLLSCAN', direction: 'forward' }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 4,
    executionTimeMillis: 126,
```

Индексы:

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

Скорость выполнения с индексами:


Наглядно видно, что запрос с индексами в разы быстрее

**Вывод:**
В ходе выполнения лабораторной работы я попрактиковал навыки работы с CRUD-операциями, с вложенными объектами в коллекции базы данных, MongoDB, агрегациями и изменением данных, с ссылками и индексами в базе данных MongoDB