

Департамент образования города Москвы  
Государственное автономное образовательное учреждение высшего  
образования города Москвы  
«Московский городской педагогический университет»

Институт цифрового образования  
Департамент информатики, управления и технологий

Инструменты для хранения и обработки больших данных  
Лабораторная работа 2.1

Изучение методов хранения данных на основе NoSQL

Выполнила: студентка группы АДЭУ-221

Пришлецова Кристина Сергеевна

Проверил:

доцент департамента информатики, управления и технологий

Босенко Тимур Муртазович

Москва

2025

**Цель работы:** изучить и практически применить три различных типа нереляционных баз данных: MongoDB, Neo4j, Redis. Создать, заполнить и проанализировать структуры данных в каждой системе, а также выполнить запросы.

**Краткое описание процесса выполнения:**

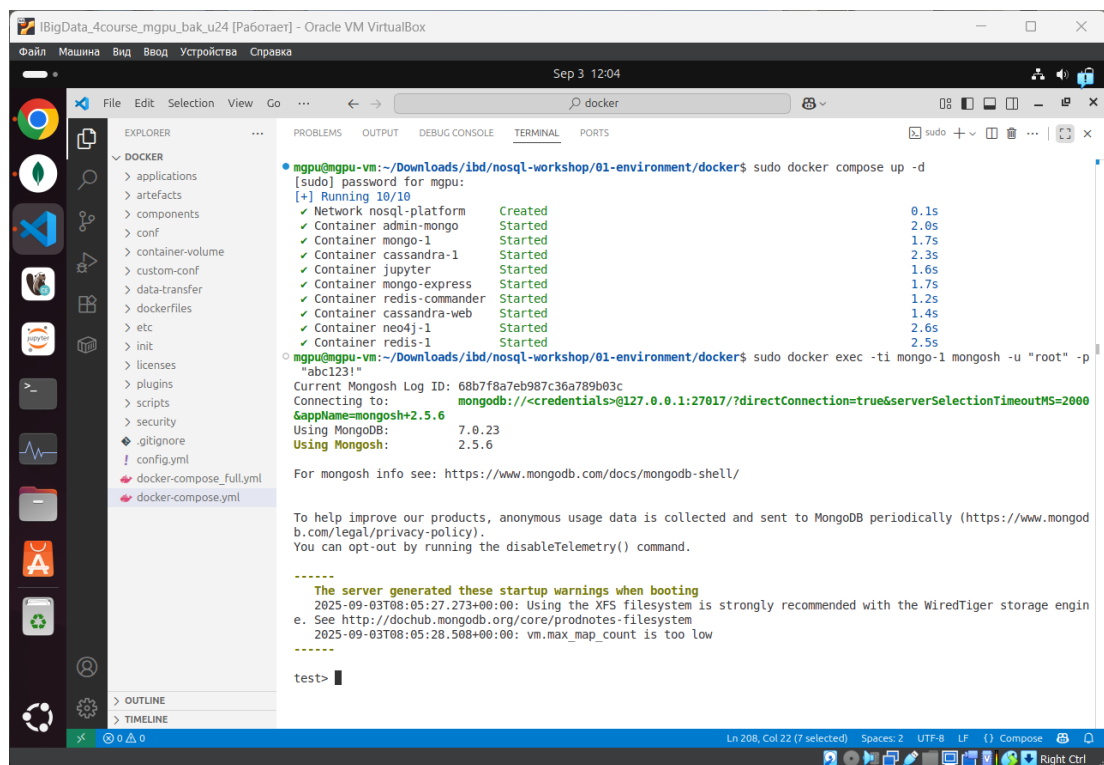
1. Подключение к базе данных MongoDB, создание коллекций, документов, заполнение документов, работа с документами, их изменение и удаление.
2. Подключение к Redis Commander, рассмотрение разных структур данных (string, set, hash и т.д.)
3. Подключение к Neo4j Browser, создание узлов, отношений, атрибутов.

## **Оглавление**

Подготовка окружения .....	3
Практическая работа 1. Создание документов в MongoDB.....	5
Задания для работы с MongoDB на Python.....	7
Задание для самостоятельной работы MongoDB.....	13
Работа с MongoDB через pymongo .....	15
Начало работы с Redis.....	16
Индивидуальное задание на Python в Redis.....	17
Задание для самостоятельной работы Redis .....	19
Работа с Redis через redis-py .....	19
Работа с Neo4J.....	20
Индивидуальное задание .....	20
Выполнение заданий .....	21
Задание для самостоятельной работы Neo4j .....	23
Работа с Neo4j через neo4j-driver.....	24

## Вариант 11.

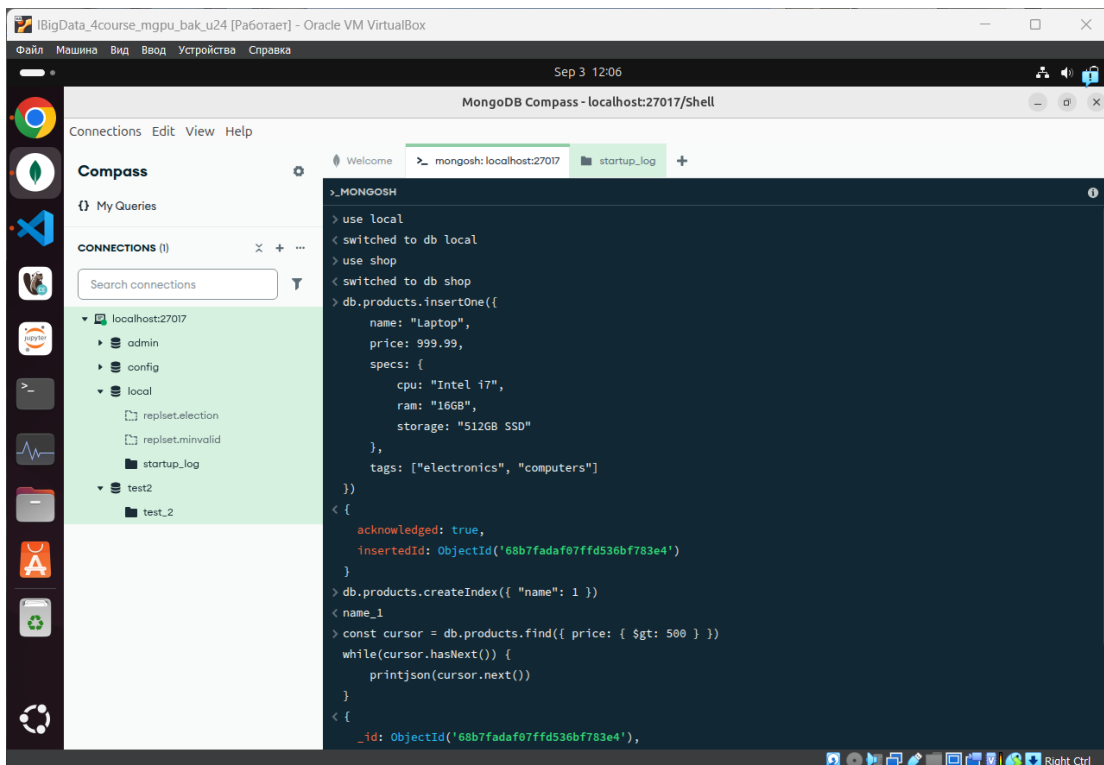
# Подготовка окружения



The screenshot shows a terminal window titled "BigData\_4course\_mgpu\_bak\_u24 [Работает] - Oracle VM VirtualBox". The terminal is running a Docker Compose command to start a set of containers. The output shows the following containers starting:

- Network nosql-platform Created 0.1s
- Container admin-mongo Started 2.0s
- Container mongo-1 Started 1.7s
- Container cassandra-1 Started 2.3s
- Container jupyter Started 1.6s
- Container mongo-express Started 1.7s
- Container redis-commander Started 1.2s
- Container cassandra-web Started 1.4s
- Container neo4j-1 Started 2.6s
- Container redis-1 Started 2.5s

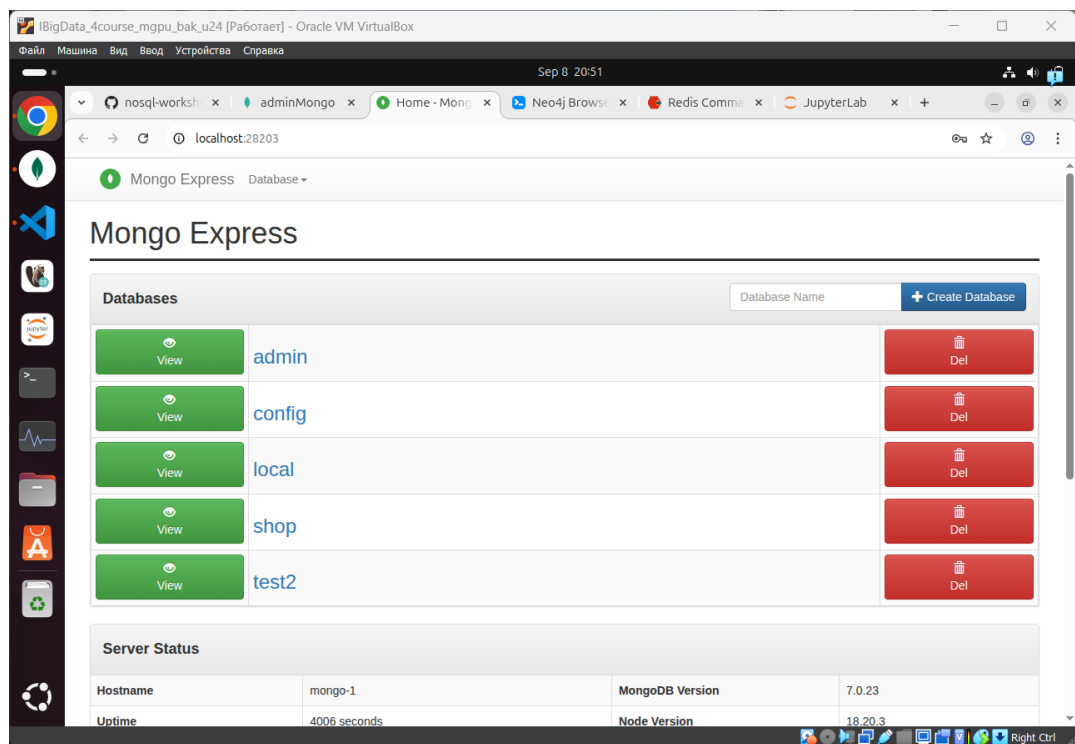
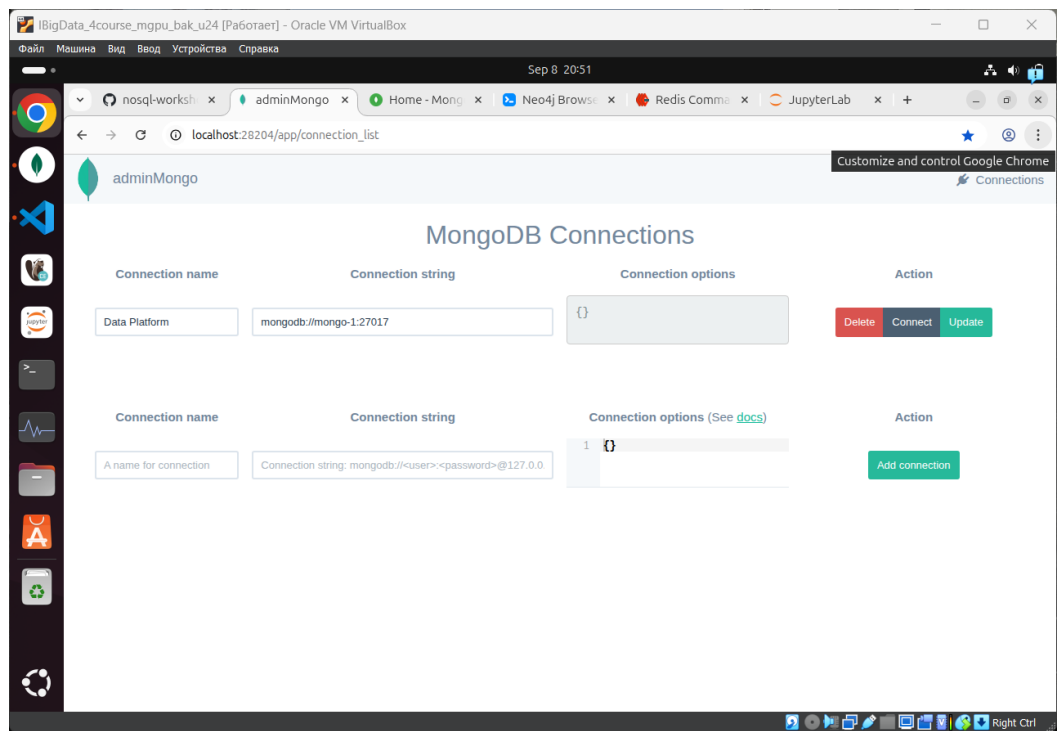
The terminal also shows the output of the `mongo` command, indicating that the MongoDB shell is running successfully. The output includes the current MongoDB Log ID, the connection string, and the version of MongoDB and Mongosh.



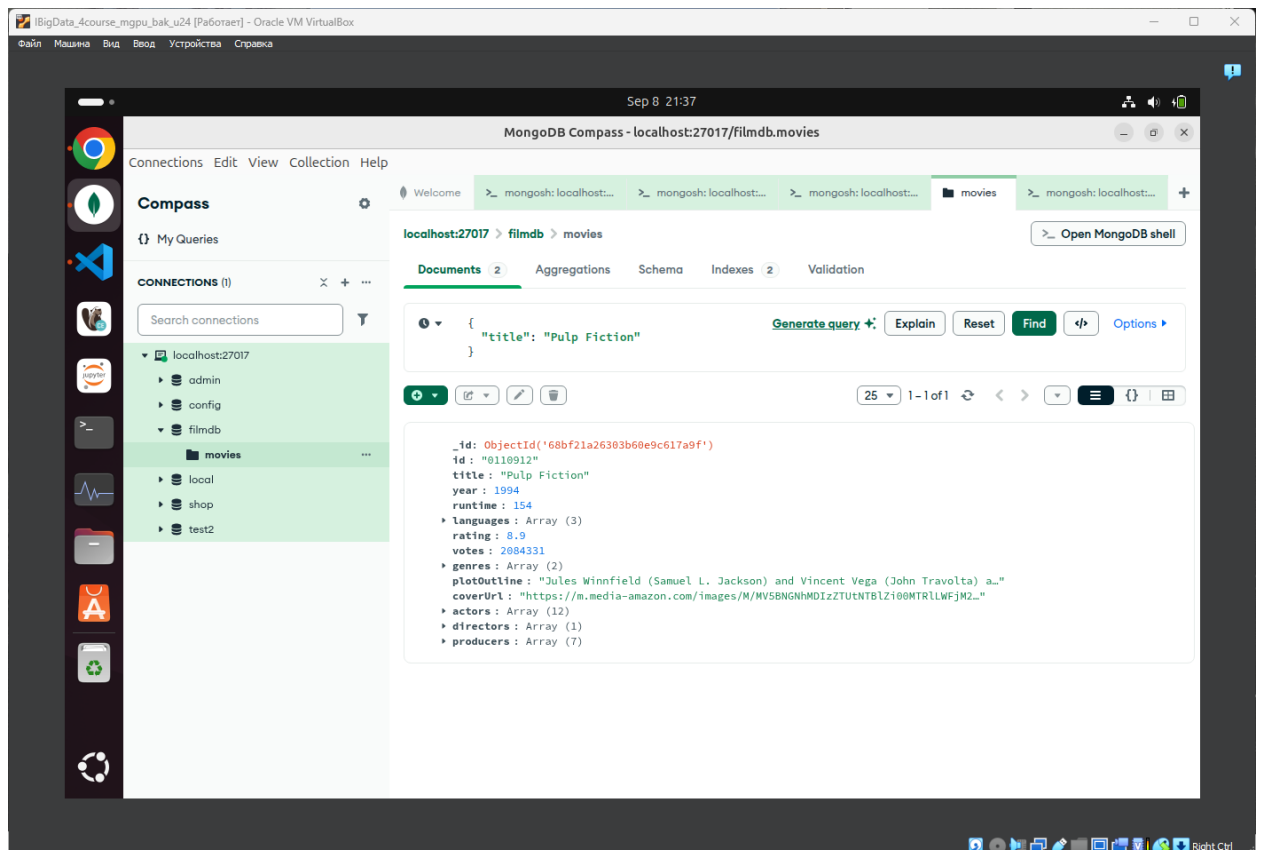
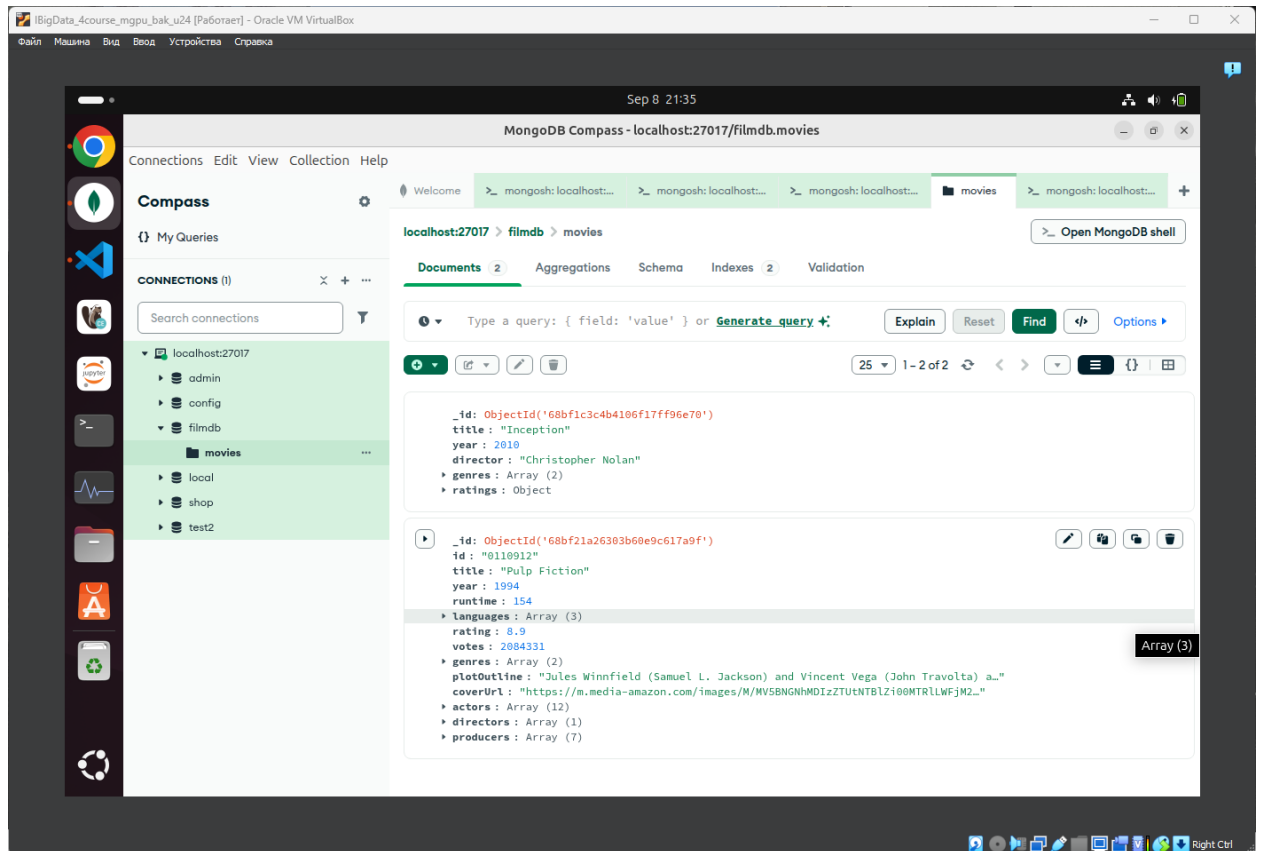
The screenshot shows the MongoDB Compass interface. The left sidebar displays the "Connections" list, with the "localhost:27017" connection selected. The "My Queries" tab is active, showing a query that filters for products with a price greater than 500. The main panel displays the results of the query, showing a single document with the following fields:

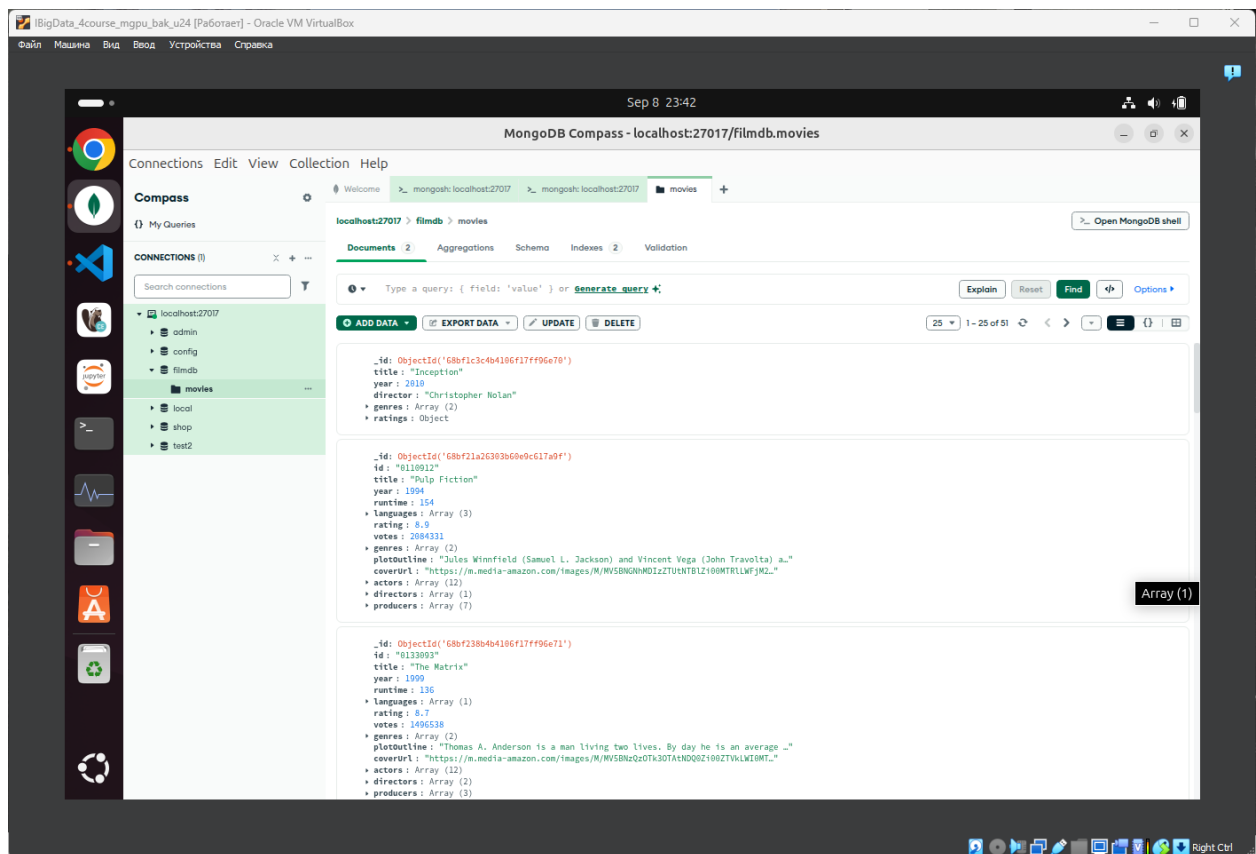
- name: "Laptop"
- price: 999.99
- specs: {
  - cpu: "Intel i7"
  - ram: "16GB"
  - storage: "512GB SSD"}
- tags: ["electronics", "computers"]

The document is acknowledged as true, and the inserted ID is shown as `ObjectId('68b7fada07fffd536bf783e4')`.



# Практическая работа 1. Создание документов в MongoDB





# Задания для работы с MongoDB на Python

## Вариант 11. Социальная сеть

Создайте базовую социальную сеть с коллекцией users

### Функционал:

- Создание пользователя

```
> use users
< switched to db users
```

```
> db.users.insertOne({
  username: "username1",
  email: "mongouser1@gmail.com",
  profile: {
    full_name: "Michael Lewia",
    bio: "Love travel and sports",
    avatar: "avatar1.png",
    location: "Moscow"
  },
  friends: ["username2", "username3", "username4"],
  posts: [
    {
      content: "Visited Sochi, incredibly beautiful!",
      timestamp: "2025-09-01T15:00:00",
      likes: ["username3"],
      comments: [
        {
          author: "username3",
          text: "Cool pictures!",
          timestamp: "2025-09-01T15:15:00"
        }
      ]
    }
  ],
  preferences: {
    privacy: "friends_only",
    notifications: true
  }
})
< {
  acknowledged: true,
  insertedId: ObjectId('68bf52977bab4d85f40925a0')
}
users>
```

- Добавление друга

```
>_MONGOSH
> db.users.updateOne(
  {username: "username1" },
  { $addToSet: { friends: "username5" } }
);
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.users.find({username: "username1"})
< {
  _id: ObjectId('68bf52977bab4d85f40925a0'),
  username: 'username1',
  email: 'mongouser1@gmail.com',
  profile: {
    full_name: 'Michael Lewia',
    bio: 'Love travel and sports',
    avatar: 'avatar1.png',
    location: 'Moscow'
  },
  friends: [
    'username2',
    'username3',
    'username4',
    'username5'
  ],
}
```

- Создание поста

```
>_MONGOSH
> db.users.updateOne(
  { username: "username1" },
  {
    $push: {
      posts: {
        content: "Visited Kazakhstan. Intresting.",
        timestamp: new Date(),
        likes: [],
        comments: []
      }
    }
  }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.users.find({username: "username1"})
< {
  _id: ObjectId('68bf52977bab4d85f40925a0'),
  username: 'username1',
  email: 'mongouser1@gmail.com',
  profile: {
    full_name: 'Michael Lewia',
    bio: 'Love travel and sports',
    avatar: 'avatar1.png',
    location: 'Moscow'
  },
  friends: [
    'username2',
    'username3',
    'username4',
  ],
}
```



```

>_MONGOSH
},
friends: [
  'username2',
  'username3',
  'username4',
  'username5'
],
posts: [
  {
    content: 'Visited Sochi, 'incredibly beautiful!',
    timestamp: '2025-09-01T15:00:00',
    likes: [
      'username3'
    ],
    comments: [
      {
        author: 'username3',
        text: 'Cool pictures!',
        timestamp: '2025-09-01T15:15:00'
      }
    ]
  },
  {
    content: 'Visited Kazakhstan. Intresting.',
    timestamp: 2025-09-08T22:28:37.609Z,
    likes: [],
    comments: []
  }
],
preferences: {
  privacy: 'friends_only',
  notifications: true
}
}
users>

```

- Лайк поста

```

>_MONGOSH
}
> db.users.updateOne(
  { username: "username1", "posts.0": { $exists: true } },
  { $addToSet: { "posts.0.likes": "username5" } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

```

> db.users.find({username: "username1"})
< {
  _id: ObjectId('68bf52977bab4d85f40925a0'),
  username: 'username1',
  email: 'mongouser1@gmail.com',
  profile: {
    full_name: 'Michael Lewia',
    bio: 'Love travel and sports',
    avatar: 'avatar1.png',
    location: 'Moscow'
  },
  friends: [
    'username2',
    'username3',
    'username4',
    'username5'
  ],
  posts: [
    {
      content: 'Visited Sochi, incredibly beautiful!',
      timestamp: '2025-09-01T15:00:00',
      likes: [
        'username3',
        'username5'
      ],
    },
  ],
}

```

- Комментирование поста

```

> db.users.updateOne(
  { username: "username1", "posts.1": { $exists: true } },
  {
    $push: {
      "posts.1.comments": {
        author: "username5",
        text: "Wonderful!",
        timestamp: new Date()
      }
    }
  }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

```

> db.users.find({username: "username1"})
< {

```

```

>_MONGOSH
]
},
{
  content: 'Visited Kazakhstan. Intresting.',
  timestamp: 2025-09-08T22:28:37.609Z,
  likes: [],
  comments: [
    {
      author: 'username5',
      text: 'Wonderful!',
      timestamp: 2025-09-09T03:41:36.446Z
    }
  ]
}
],
preferences: {
  privacy: 'friends_only',
  notifications: true
},
psts: {
  '0': {
    likes: [
      'username4',
      'username5'
    ]
  }
}
}
}
users>

```

- Поиск пользователей по интересам  
(создаем сначала еще одного пользователя)

```

>_MONGOSH
> db.users.insertOne({
  username: "username2",
  email: "mongouser2@gmail.com",
  profile: {
    full_name: "Terry McDonald",
    bio: "sport is life",
    avatar: "avatar2.png",
    location: "Los Angeles"
  },
  friends: ["username5", "username1"]
})
< {
  acknowledged: true,
  insertedId: ObjectId('68bfa3ceed6db41e62197c0d')
}
> db.users.find({username: "username2"})
< {
  _id: ObjectId('68bfa3ceed6db41e62197c0d'),
  username: 'username2',
  email: 'mongouser2@gmail.com',
  profile: {
    full_name: 'Terry McDonald',
    bio: 'sport is life',
    avatar: 'avatar2.png',
    location: 'Los Angeles'
  },
  friends: [
    'username5',
    'username1'
  ]
}
users>

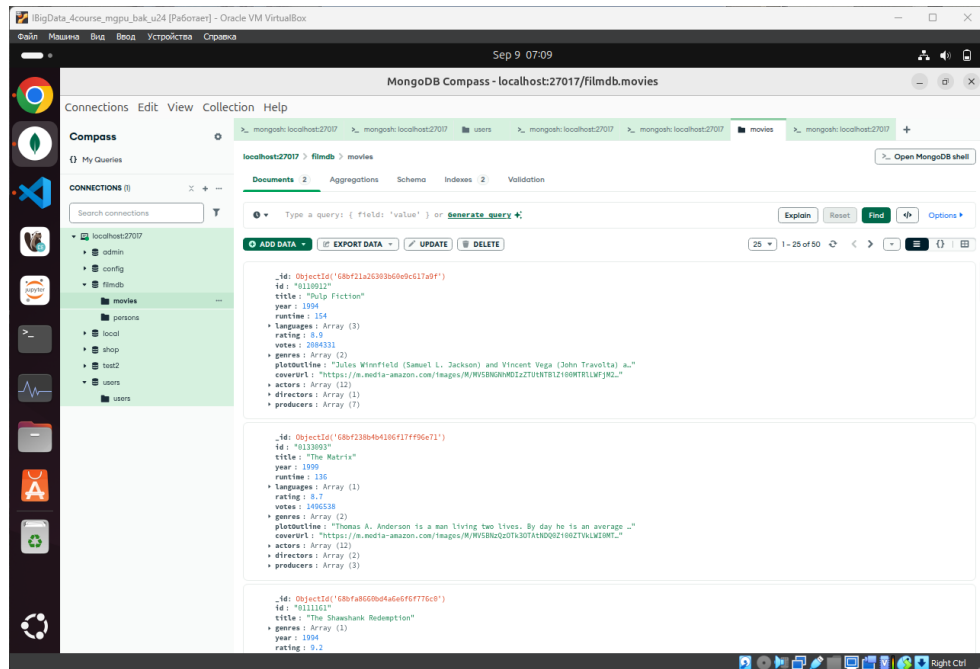
```

```
>_MONGOSH
>
> db.users.find({ "profile.bio": /sport/i}).toArray()
< [
  {
    _id: ObjectId('68bf52977bab4d85f40925a0'),
    username: 'username1',
    email: 'mongouser1@maol.com',
    profile: {
      full_name: 'Michael Lewia',
      bio: 'Love travel and sports',
      avatar: 'avatar1.png',
      location: 'Moscow'
    },
    friends: [ 'username2', 'username3', 'username4', 'username5' ],
    posts: [ [Object], [Object] ],
    preferences: { privacy: 'friends_only', notifications: true },
    psts: { '0': [Object] }
  },
  {
    _id: ObjectId('68bfa3ceed6db41e62197c0d'),
    username: 'username2',
    email: 'mongouser2@gmail.com',
    profile: {
      full_name: 'Terry McDonald',
      bio: 'sport is life',
      avatar: 'avatar2.png',
      location: 'Los Angeles'
    },
    friends: [ 'username5', 'username1' ]
  }
]
users>|
```

# Задание для самостоятельной работы MongoDB

## №1 Задание 1 (MongoDB)

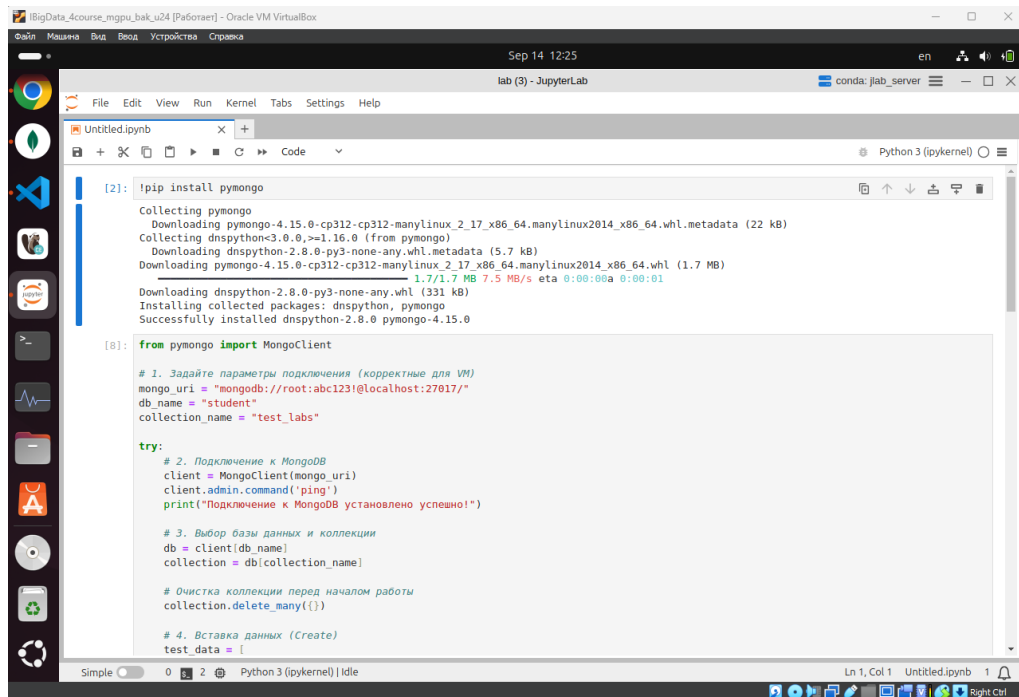
Найти все фильмы, где в массиве genres есть и "Action", и "Thriller" (\$all).



>\_MONGOSH

```
    rating: 8.5,  
    rank: 49  
  }  
> db.movies.find({"genres": { $all: ["Action", "Thriller"] } })  
< {  
  _id: ObjectId('68bfa8660bd4a6e6f6f776c3'),  
  id: '0468569',  
  title: 'The Dark Knight',  
  genres: [  
    'Action',  
    'Crime',  
    'Drama',  
    'Thriller'  
  ],  
  year: 2008,  
  rating: 9,  
  rank: 4  
}  
{  
  _id: ObjectId('68bfa8660bd4a6e6f6f776cd'),  
  id: '1375666',  
  title: 'Inception',  
  genres: [  
    'Action',  
    'Adventure',  
    'Sci-Fi',  
    'Thriller'  
  ],  
  year: 2010,  
  rating: 8.7,  
  rank: 15  
}  
{  
  _id: ObjectId('68bfa8660bd4a6e6f6f776db'),  
  id: '0110413',  
  title: 'Léon: The Professional',  
  genres: [  
    'Action',  
    'Crime',  
    'Drama',  
    'Thriller'  
  ],  
  year: 1994,  
  rating: 8.5,  
  rank: 38  
}  
filmdb>
```

# Работа с MongoDB через pymongo



```
[2]: !pip install pymongo

Collecting pymongo
  Downloading pymongo-4.15.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (22 kB)
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.8.0-py3-none-any.whl.metadata (5.7 kB)
Downloaded dnspython-2.8.0-py3-none-any.whl (1.7 MB)
1.7/1.7 MB 7.5 MB/s eta 0:00:00a 0:00:01
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.8.0 pymongo-4.15.0

[8]: from pymongo import MongoClient

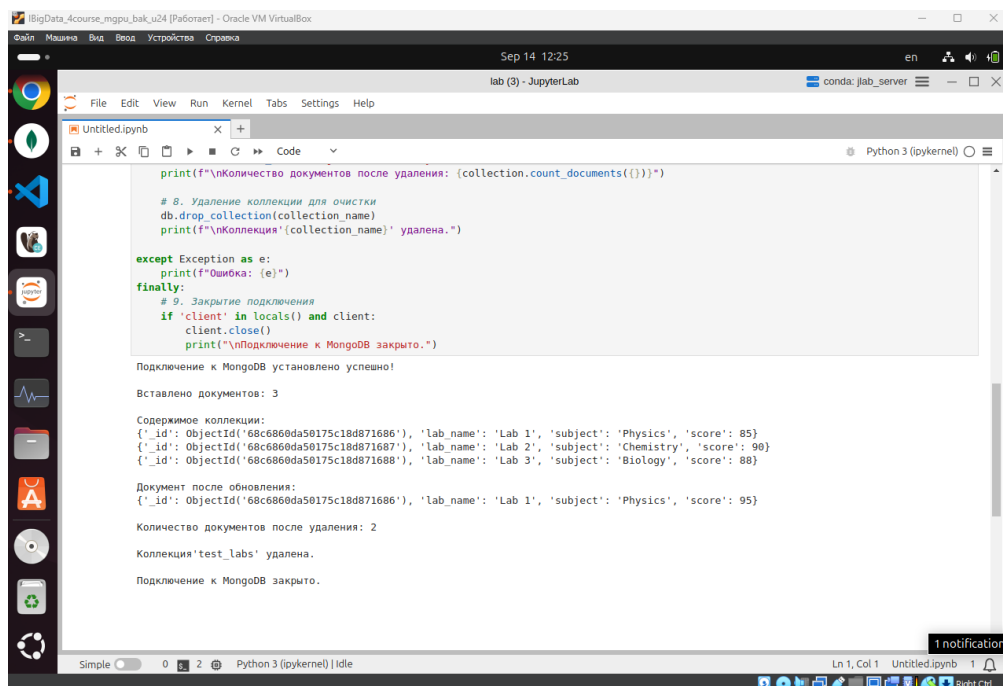
# 1. Задайте параметры подключения (корректные для VM)
mongo_uri = "mongodb://root:abc123!@localhost:27017/"
db_name = "student"
collection_name = "test_labs"

try:
    # 2. Подключение к MongoDB
    client = MongoClient(mongo_uri)
    client.admin.command('ping')
    print("Подключение к MongoDB установлено успешно!")

    # 3. Выбор базы данных и коллекции
    db = client[db_name]
    collection = db[collection_name]

    # 4. Очистка коллекции перед началом работы
    collection.delete_many({})

    # 5. Вставка данных (Create)
    test_data = [
```



```
print(f"\nКоличество документов после удаления: {collection.count_documents({})}")

# 8. Удаление коллекции для очистки
db.drop_collection(collection_name)
print(f"\nКоллекция '{collection_name}' удалена.")

except Exception as e:
    print(f"Ошибка: {e}")
finally:
    # 9. Закрытие подключения
    if 'client' in locals() and client:
        client.close()
    print("\nПодключение к MongoDB закрыто.")

Подключение к MongoDB установлено успешно!

Вставлено документов: 3

Содержимое коллекции:
{'_id': ObjectId('68c6860da50175c18d871686'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 85}
{'_id': ObjectId('68c6860da50175c18d871687'), 'lab_name': 'Lab 2', 'subject': 'Chemistry', 'score': 90}
{'_id': ObjectId('68c6860da50175c18d871688'), 'lab_name': 'Lab 3', 'subject': 'Biology', 'score': 88}

Документ после обновления:
{'_id': ObjectId('68c6860da50175c18d871686'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 95}

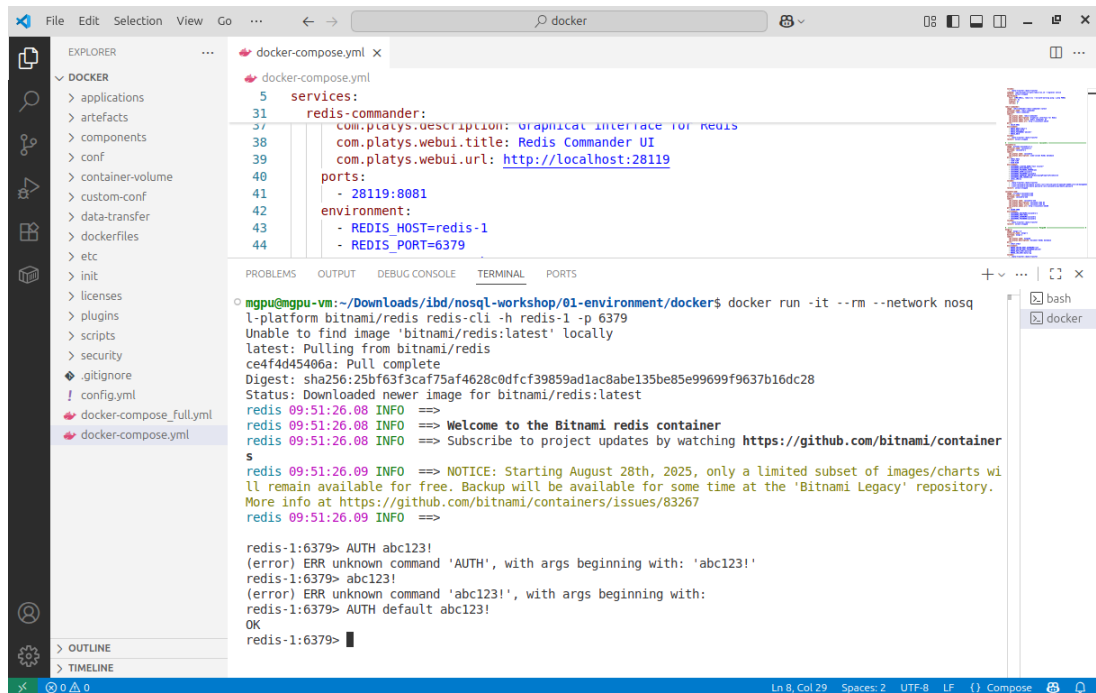
Количество документов после удаления: 2

Коллекция 'test_labs' удалена.

Подключение к MongoDB закрыто.
```

# Начало работы с Redis

## Запуск Redis CLI

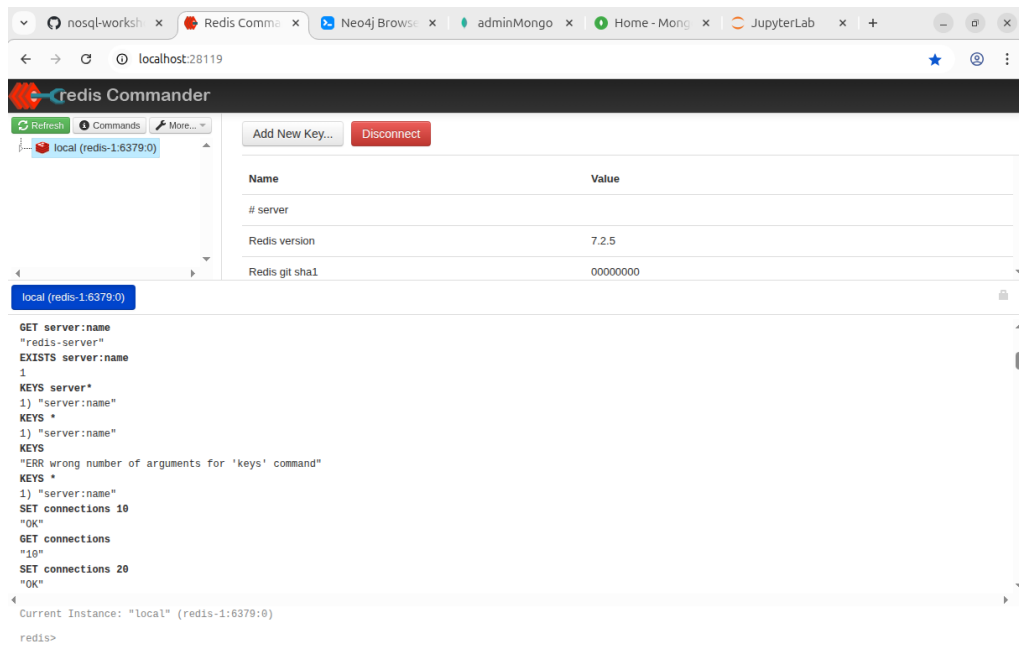


The screenshot shows a VS Code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with folders like 'applications', 'artefacts', 'components', 'conf', 'container-volume', 'custom-conf', 'data-transfer', 'dockerfiles', 'etc', 'init', 'licenses', 'plugins', 'scripts', 'security', and files like '.gitignore', 'config.yml', 'docker-compose\_full.yml', and 'docker-compose.yml'. The 'docker-compose.yml' file is open in the editor, showing the following configuration:

```
5 services:
31   redis-commander:
32     com.platys.description: Graphical interface for Redis
38     com.platys.webui.title: Redis Commander UI
39     com.platys.webui.url: http://localhost:28119
40   ports:
41     - 28119:8081
42   environment:
43     - REDIS_HOST=redis-1
44     - REDIS_PORT=6379
```

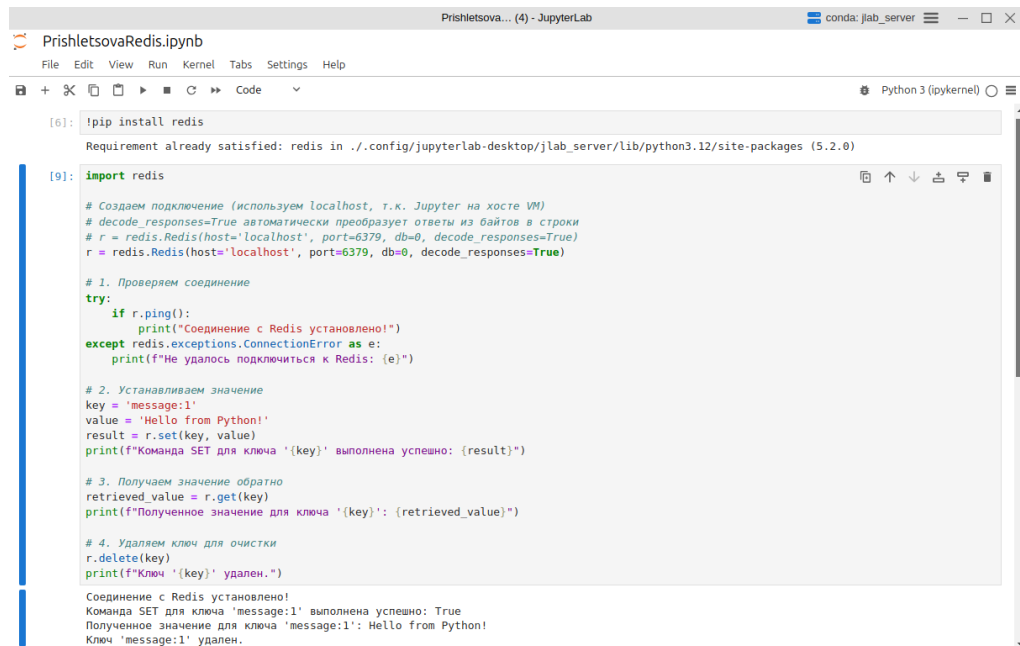
The terminal shows the command `docker run -it --rm --network nosql-platform bitnami/redis redis-cli -h redis-1 -p 6379` being executed. The output shows that the image 'bitnami/redis:latest' is pulled from Docker Hub. The terminal then shows the Redis CLI prompt `redis-1:6379>` and the command `AUTH abc123!` being executed, resulting in an error: `(error) ERR unknown command 'AUTH', with args beginning with: 'abc123!'`. The terminal also shows the command `SET connections 10` being executed, resulting in `"OK"`.

## Запуск Redis Commander





## Python-пример



```
Prishletsova... (4) - JupyterLab
conda: jlab_server

PrishletsovaRedis.ipynb
File Edit View Run Kernel Tabs Settings Help

Python 3 (ipykernel)

[6]: !pip install redis
Requirement already satisfied: redis in ./config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (5.2.0)

[9]: import redis

# Создаем подключение (используем localhost, т.к. Jupyter на хосте VM)
# decode_responses=True автоматически преобразует ответы из байтов в строки
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)

# 1. Проверяем соединение
try:
    if r.ping():
        print("Соединение с Redis установлено!")
except redis.exceptions.ConnectionError as e:
    print(f"Не удалось подключиться к Redis: {e}")

# 2. Устанавливаем значение
key = 'message:1'
value = 'Hello from Python!'
result = r.set(key, value)
print(f"Команда SET для ключа '{key}' выполнена успешно: {result}")

# 3. Получаем значение обратно
retrieved_value = r.get(key)
print(f"Полученное значение для ключа '{key}': {retrieved_value}")

# 4. Удаляем ключ для очистки
r.delete(key)
print(f"Ключ '{key}' удален.")

Соединение с Redis установлено!
Команда SET для ключа 'message:1' выполнена успешно: True
Полученное значение для ключа 'message:1': Hello from Python!
Ключ 'message:1' удален.
```

## Индивидуальное задание на Python в Redis

### Вариант 1, 11, 21 Система счетчиков посещений

Создайте систему подсчета посещений веб-страниц. Реализуйте функции для:

- Увеличения счетчика посещений страницы
- Получения количества посещений
- Получения топ-5 самых посещаемых страниц
- Сброса статистики

# Пример структуры данных

# page\_views:home = 150

# page\_views:about = 75

```
PrishletsovaRedis.ipynb
File Edit View Run Kernel Tabs Settings Help

Python 3 (ipykernel)

[12]: import redis

# Создаем подключение (используем localhost, т.к. Jupyter на хосте VM)
# decode_responses=True автоматически преобразует ответы из байтов в строки
# r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)

# 1. Проверяем соединение
try:
    if r.ping():
        print("Соединение с Redis установлено!")
except redis.exceptions.ConnectionError as e:
    print(f"Не удалось подключиться к Redis: {e}")

# 2. Увеличение счетчика посещений страницы
def increment_page_view(page_name: str):
    key = f"page_views:{page_name}"
    return r.incr(key) # увеличивает значение на 1 и возвращает новое

# 3. Получение количества посещений страницы
def get_page_views(page_name: str):
    key = f"page_views:{page_name}"
    value = r.get(key)
    return int(value) if value else 0

# 4. Получение топ-5 самых посещаемых страниц
def get_top_pages(limit=5):
    keys = r.keys("page_views:*")
    stats = []
    for key in keys:
        page = key.split(":")[1]
        views = int(r.get(key))
        stats.append((page, views))
    # сортируем по убыванию
    stats.sort(key=lambda x: x[1], reverse=True)
    return stats[:limit]

# 5. Сброс статистики
def reset_stats():
    keys = r.keys("page_views:*")
    if keys:
        r.delete(*keys)


```

```
PrishletsovaRedis.ipynb
File Edit View Run Kernel Tabs Settings Help

Python 3 (ipykernel)

def reset_stats():
    keys = r.keys("page_views:*")
    if keys:
        r.delete(*keys)

reset_stats()

increment_page_view("home")
increment_page_view("home")
increment_page_view("home")
increment_page_view("about")
increment_page_view("about")
increment_page_view("contact")
increment_page_view("about")
increment_page_view("home")
increment_page_view("home")
increment_page_view("about")
increment_page_view("download")
increment_page_view("gallery")

print("home views:", get_page_views("home"))
print("about views:", get_page_views("about"))
print("contact views:", get_page_views("contact"))
print("download views:", get_page_views("download"))
print("gallery views:", get_page_views("gallery"))

print("TOP pages:", get_top_pages())

reset_stats()
print("После сброса:", get_top_pages())

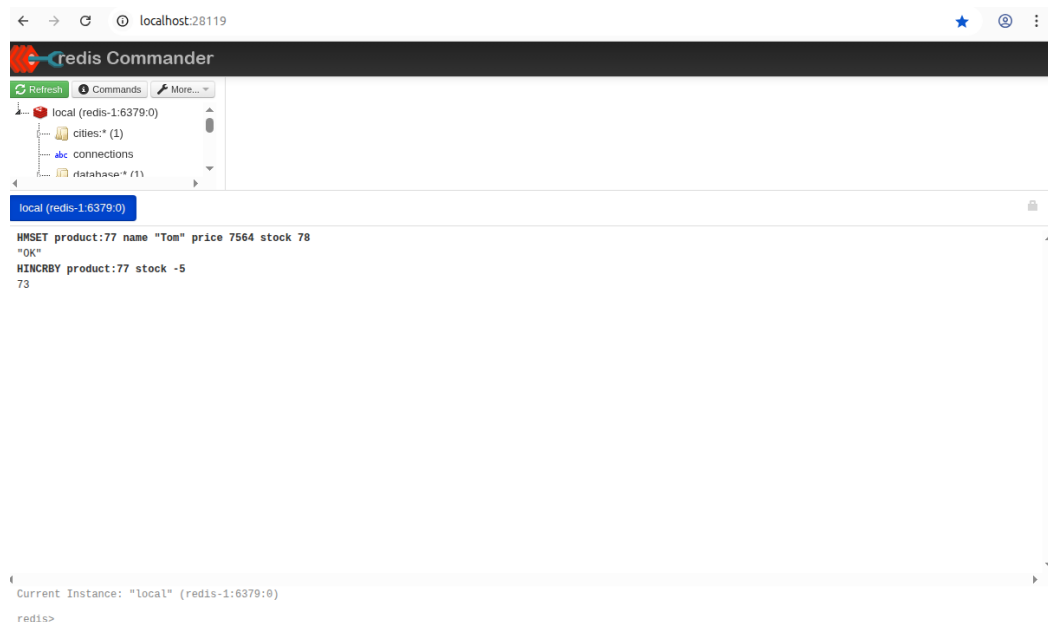
Соединение с Redis установлено!
home views: 5
about views: 4
contact views: 1
download views: 1
gallery views: 1
TOP pages: [('home', 5), ('about', 4), ('download', 1), ('contact', 1), ('gallery', 1)]
После сброса: []


```

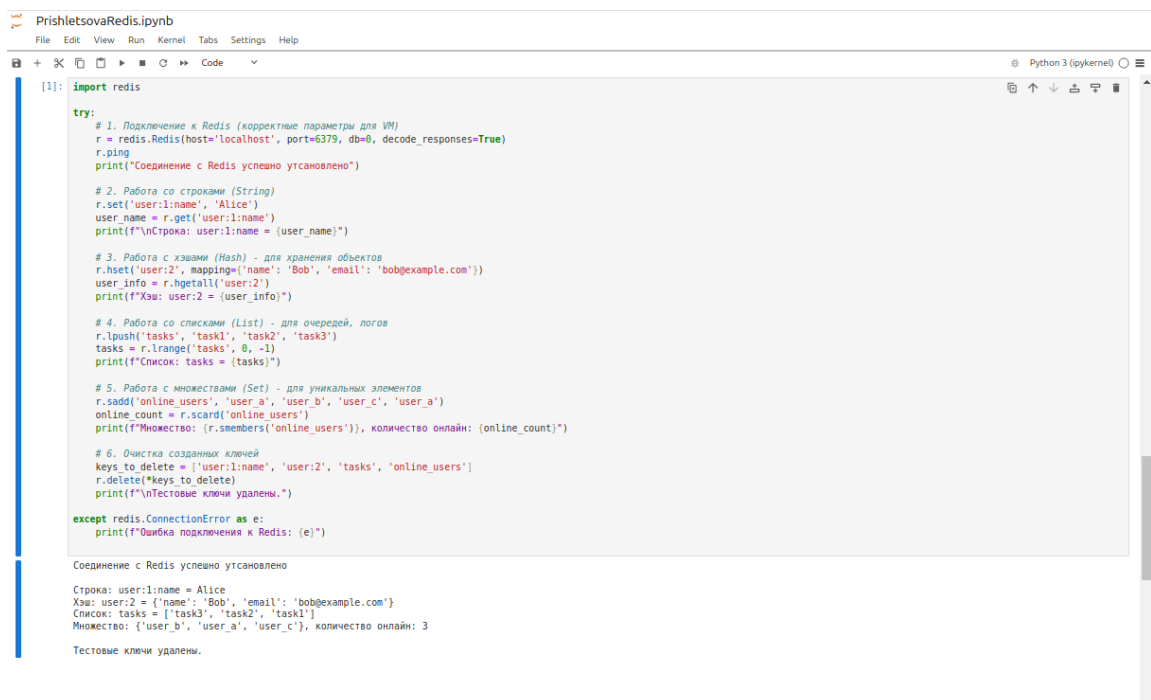
# Задание для самостоятельной работы Redis

## №2 Задание 3 (Redis)

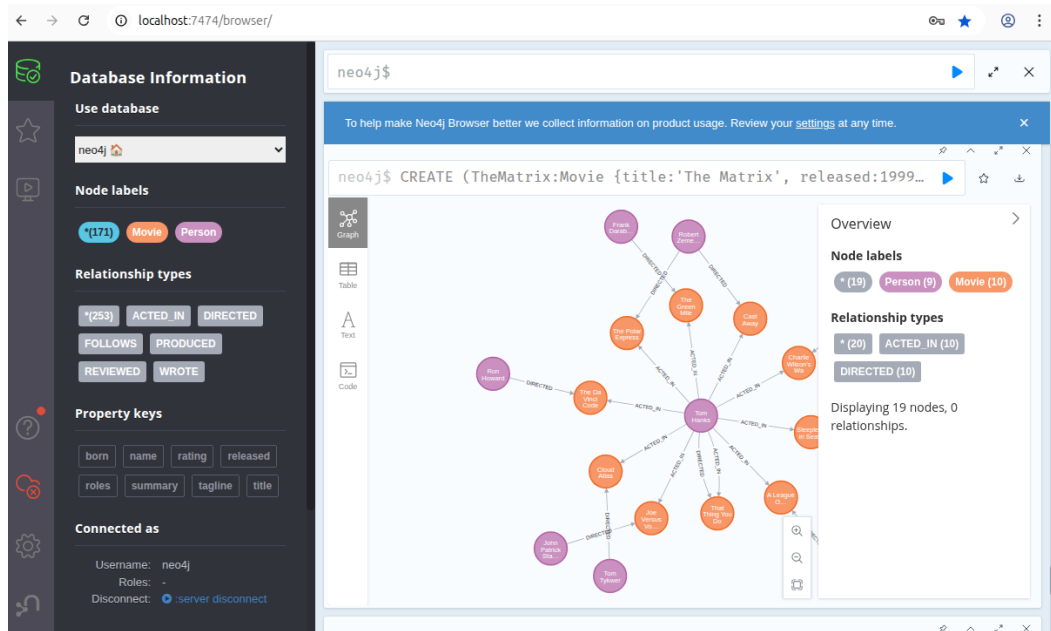
Создать хэш product:77 с полями name, price, stock. Уменьшить значение stock на 5 (HINCRBY с отрицательным значением).



## Работа с Redis через redis-py



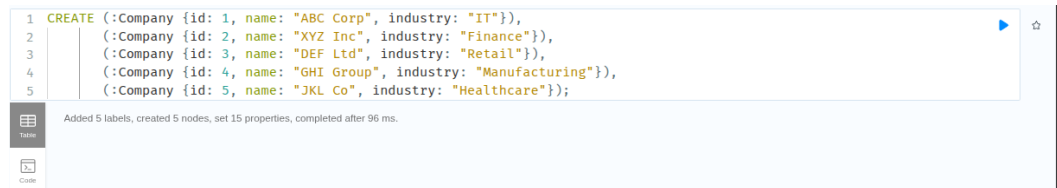
# Работа с Neo4J



## Индивидуальное задание

1. Создание графовой базы данных на основе предоставленных данных о компаниях, сотрудниках, проектах и клиентах.

### Создание компаний



### Создание сотрудников



### Создание проектов



## Создание клиентов

```
1 CREATE (:Client {id: 1, name: "Client A", project_id: 1}),
2       (:Client {id: 2, name: "Client B", project_id: 2}),
3       (:Client {id: 3, name: "Client C", project_id: 3}),
4       (:Client {id: 4, name: "Client D", project_id: 4}),
5       (:Client {id: 5, name: "Client E", project_id: 5}),
6       (:Client {id: 6, name: "Client F", project_id: 6}),
7       (:Client {id: 7, name: "Client G", project_id: 7}),
8       (:Client {id: 8, name: "Client H", project_id: 8}),
9       (:Client {id: 9, name: "Client I", project_id: 9}),
10      (:Client {id: 10, name: "Client J", project_id: 10});
```

Added 10 labels, created 10 nodes, set 30 properties, completed after 72 ms.

## Создание отношений

```
1 MATCH (e:Employee), (c:Company)
2 WHERE e.company_id = c.id
3 CREATE (e)-[:WORKS_AT]-(c);
4
5 MATCH (p:Project), (c:Company)
6 WHERE p.company_id = c.id
7 CREATE (c)-[:HAS_PROJECT]-(p);
8
9 MATCH (e:Employee), (p:Project)
10 WHERE (e.id = 1 AND p.id = 1) OR (e.id = 2 AND p.id = 2) OR (e.id = 3 AND p.id = 3) OR (e.id = 4 AND p.id = 4) OR
11        (e.id = 5 AND p.id = 5)
12 CREATE (e)-[:PARTICIPATES_IN {role: CASE WHEN e.id = 1 THEN "Lead Developer" WHEN e.id = 2 THEN "Project Manager"
13        ]-(p);
```

neo4j\$ MATCH (e:Employee), (c:Company) WHERE e.company\_id = c.id CREATE (e)-[:WORKS\_AT]-(c) ✓

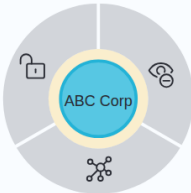
neo4j\$ MATCH (p:Project), (c:Company) WHERE p.company\_id = c.id CREATE (c)-[:HAS\_PROJECT]-(p) ✓

neo4j\$ MATCH (e:Employee), (p:Project) WHERE (e.id = 1 AND p.id = 1) OR (e.id = 2 AND p.id = 2) OR (e.id = 3 AND p.id = 3) OR (e.id = 4 AND p.id = 4) OR (e.id = 5 AND p.id = 5) CREATE (e)-[:PARTICIPATES\_IN {role: CASE WHEN e.id = 1 THEN "Lead Developer" WHEN e.id = 2 THEN "Project Manager"}]-(p) ✓

## Выполнение заданий

Задание 1. Найдите все компании из сферы технологий.

```
1 MATCH (c:Company)
2 WHERE c.industry = "IT"
3 RETURN c;
```



Node properties

Property	Value
id	4e4cc1a96-4093-44d8-8102-23a43ac8f493:171
id	171
id	1
industry	IT
name	ABC Corp

Задание 2. Найдите всех сотрудников, занимающих должность "Lead Developer".

```
1 MATCH (e:Employee)-[:PARTICIPATES_IN]-(p:Project)
2 WHERE r.role = "Lead Developer"
3 RETURN e.name AS employee_name, e.position, p.name AS project_name;
```

employee_name	e.position	project_name
"John Doe"	"Developer"	"Project X"

Задание 3. Найдите все проекты, продолжительностью от 4 до 6 месяцев.

```
1 MATCH (p:Project)
2 WITH p, date(p.start_date) AS start, date(p.end_date) AS finish
3 WHERE duration.inDays(start, finish).days >= 120
4 | AND duration.inDays(start, finish).days <= 180
5 RETURN p.name AS project_name, p.start_date, p.end_date
```

	project_name	p.start_date	p.end_date
1	"Project X"	"2023-01-01"	"2023-06-01"
2	"Project Y"	"2023-02-01"	"2023-07-01"
3	"Project Z"	"2023-03-01"	"2023-08-01"
4	"Project A"	"2023-04-01"	"2023-09-01"
5	"Project B"	"2023-05-01"	"2023-10-01"
6	"Project C"	"2023-06-01"	"2023-11-01"
7	"Project D"	"2023-07-01"	"2023-12-01"
8	"Project E"	"2023-08-01"	"2024-01-01"

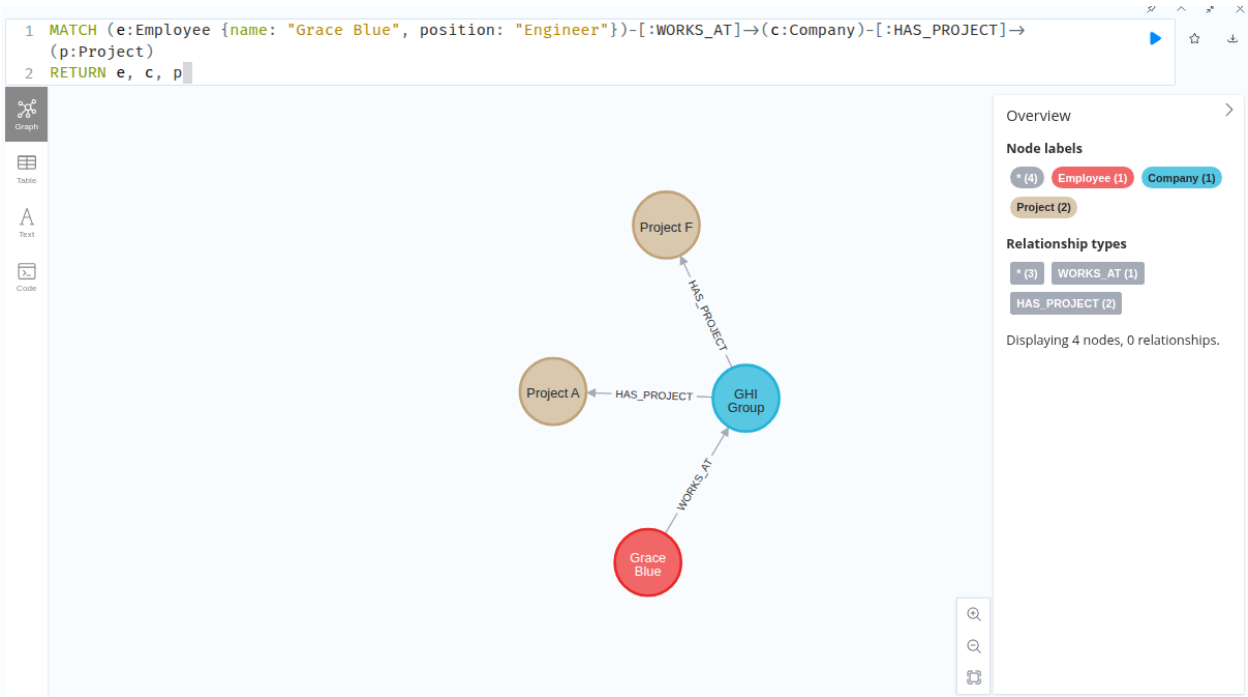
Started streaming 10 records after 12 ms and completed after 15 ms.

Задание 4. Найдите всех клиентов проекта "Project F".

```
neo4j$ MATCH (c:Client)-[:IS_CLIENT_OF]-(p:Project {name: "Project F"}) RETURN c.name AS client_name;
```

	client_name
1	"Client I"

Задание 5. Найдите все проекты, в которых сотрудник "Grace Blue" является инженером.



# Задание для самостоятельной работы Neo4j

## №3 Задание 2 (Neo4j)

Найти всех людей, которые связаны с фильмом "Cloud Atlas" (актеры, режиссеры и т.д.), и тип их связи.

```
neo4j$ MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
```

	people.name	Type(relatedTo)	relatedTo
1	"Tom Hanks"	"ACTED_IN"	{ "identity": 137, "start": 71, "end": 105, "type": "ACTED_IN", "properties": { "roles": [ "Zachry", "Dr. Henry Goose", "Isaac Sachs", "Dermot Hoggins" ] }, "elementId": "5:e4cc1a96-4093-44d8-8102-23a43ac8f493:137", "startNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:71", "endNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:105" }
2	"Jim Broadbent"	"ACTED_IN"	{ "identity": 140, "start": 107, "end": 105, "type": "ACTED_IN", "properties": { "roles": [ "Zachry", "Dr. Henry Goose", "Isaac Sachs", "Dermot Hoggins" ] }, "elementId": "5:e4cc1a96-4093-44d8-8102-23a43ac8f493:140", "startNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:107", "endNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:105" }

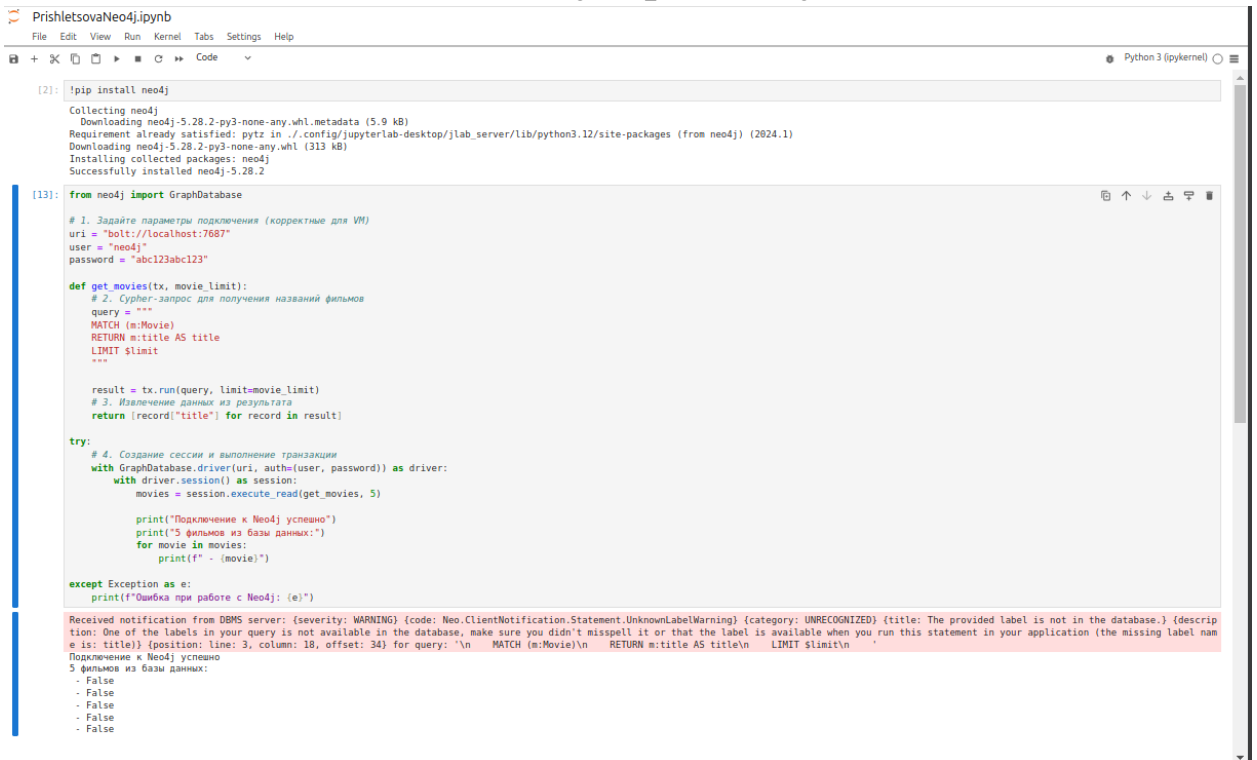
Started streaming 10 records after 1 ms and completed after 2 ms.

```
neo4j$ MATCH (people:Person)-[relatedTo]-(:Movie {title: "Cloud Atlas"}) RETURN people.name, Type(relatedTo), relatedTo
```

	people.name	Type(relatedTo)	relatedTo
3	"David Mitchell"	"WROTE"	{ "identity": 144, "start": 109, "end": 105, "type": "WROTE", "properties": { "roles": [ "Zachry", "Dr. Henry Goose", "Isaac Sachs", "Dermot Hoggins" ] }, "elementId": "5:e4cc1a96-4093-44d8-8102-23a43ac8f493:144", "startNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:109", "endNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:105" }
4	"Tom Tykwer"	"DIRECTED"	{ "identity": 141, "start": 108, "end": 105, "type": "DIRECTED", "properties": { "roles": [ "Zachry", "Dr. Henry Goose", "Isaac Sachs", "Dermot Hoggins" ] }, "elementId": "5:e4cc1a96-4093-44d8-8102-23a43ac8f493:141", "startNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:108", "endNodeElementId": "4:e4cc1a96-4093-44d8-8102-23a43ac8f493:105" }

Started streaming 10 records after 1 ms and completed after 2 ms.

# Работа с Neo4j через neo4j-driver



```
[2]: !pip install neo4j

Collecting neo4j
  Downloading neo4j-5.28.2-py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: pytz in ./.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from neo4j) (2024.1)
  Downloading neo4j-5.28.2-py3-none-any.whl (313 kB)
Installing collected packages: neo4j
Successfully installed neo4j-5.28.2

[13]: from neo4j import GraphDatabase

# 1. Задайте параметры подключения (корректные для VM)
uri = "bolt://localhost:7687"
user = "neo4j"
password = "abc123abc123"

def get_movies(tx, movie_limit):
    # 2. Cypher-запрос для получения названий фильмов
    query = """
    MATCH (m:Movie)
    RETURN m:title AS title
    LIMIT $limit
    """

    result = tx.run(query, limit=movie_limit)
    # 3. Извлечение данных из результата
    return [record["title"] for record in result]

try:
    # 4. Создание сессии и выполнение транзакции
    with GraphDatabase.driver(uri, auth=(user, password)) as driver:
        with driver.session() as session:
            movies = session.execute_read(get_movies, 5)

            print("Подключение к Neo4j успешно")
            print("5 фильмов из базы данных:")
            for movie in movies:
                print(f" - {movie}")

except Exception as e:
    print(f"Ошибка при работе с Neo4j: {e}")

Received notification from DBMS server: (severity: WARNING) (code: Neo.ClientNotification.Statement.UnknownLabelWarning) (category: UNRECOGNIZED) (title: The provided label is not in the database.) (description: One of the labels in your query is not available in the database, make sure you didn't misspell it or that the label is available when you run this statement in your application (the missing label name is: title)) (position: line: 3, column: 18, offset: 34) for query: '\n MATCH (m:Movie)\n RETURN m:title AS title\n LIMIT $limit\n '
Подключение к Neo4j успешно
5 фильмов из базы данных:
- False
- False
- False
- False
- False
```

## Выводы:

В работе были изучены три типа нереляционных СУБД: MongoDB, Redis и Neo4j.

В MongoDB освоены основные операции с коллекциями и документами, в Redis — работа с различными структурами данных (string, set, hash и др.), в Neo4j — создание узлов, связей и выполнение запросов на языке Cypher.

Сложности возникали при подключении и написании запросов, но они были решены настройкой пароля и корректировкой синтаксиса.

В результате получены практические навыки работы с документно-ориентированными, графовыми и key-value базами данных, что позволило понять их особенности и области применения.