Департамент образования города Москвы

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

Институт цифрового образования
Департамент информатики, управления и технологий

ПРАКТИЧЕСКАЯ РАБОТА №1

по дисциплине «Инструменты для хранения и обработки больших данных»
Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
Студентка группы АДЭУ-221
Вознесенская В. Е.

Проверил:
Босенко Т. М., доцент

Москва
2025

```
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ sudo  docker compose up -d
[sudo] password for mgpu:
[+] Running 10/10
 ✔ Network nosql-platform      Created                                    0.2s
 ✔ Container admin-mongo       Started                                    7.1s
 ✔ Container jupyter           Started                                    6.1s
 ✔ Container cassandra-web     Started                                    7.3s
 ✔ Container cassandra-1       Started                                    6.7s
 ✔ Container mongo-1           Started                                    4.4s
 ✔ Container redis-commander   Started                                    4.9s
 ✔ Container neo4j-1           Started                                    7.8s
 ✔ Container redis-1           Started                                    4.5s
 ✔ Container mongo-express     Started                                    4.7s
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$
                                        Ln 208, Col 22 (7 selected)  Spaces: 2  UTF-8  LF  {} Compose
```

Запуск sudo docker compose up -d

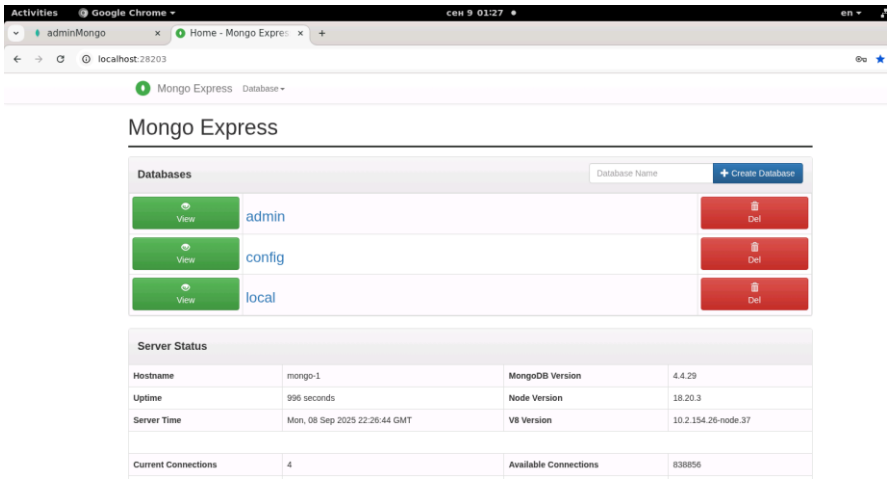
sudo docker compose stop

```
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ sudo docker exec -it mongo-1 mongo -u root -p abc123! --authenticationDatabase ad
min
[sudo] password for mgpu:
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("9a7b7f36-2968-4293-a87b-c66486f32f6f") }
MongoDB server version: 4.4.29
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2025-09-08T22:10:08.820+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongod
b.org/core/prodnotes-filesystem
---
>
```

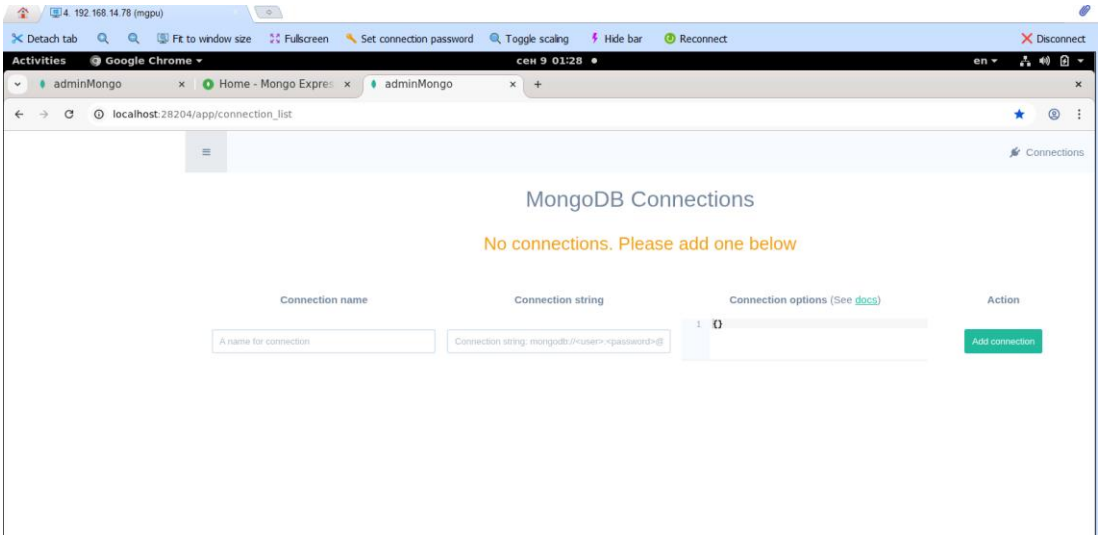
(остановка контейнера sudo docker stop mongo-1)

Теперь вы находитесь в командной строке MongoDB, готовой к выполнению любых операторов MongoDB. Также можно увидеть версию сервера MongoDB и оболочки MongoDB.

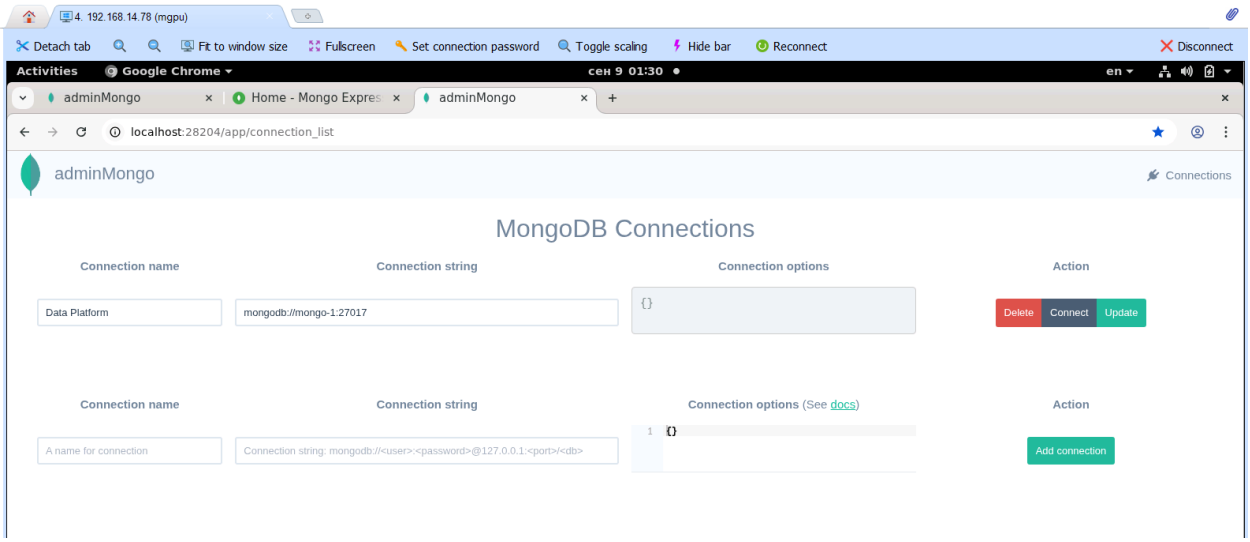
Команда: sudo docker exec -it mongo-1 mongo -u root -p abc123! --authenticationDatabase admin



http://localhost:28203/



http://localhost:28204/



Добавили подключение

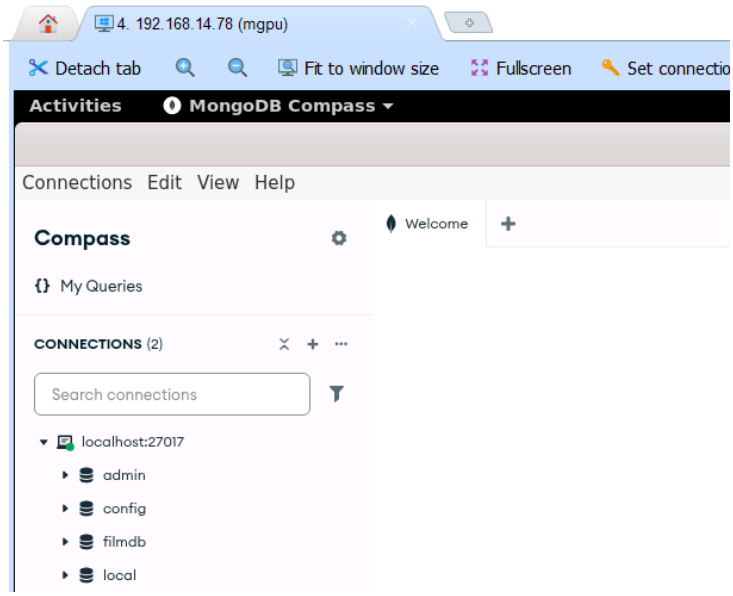
Практический Пример

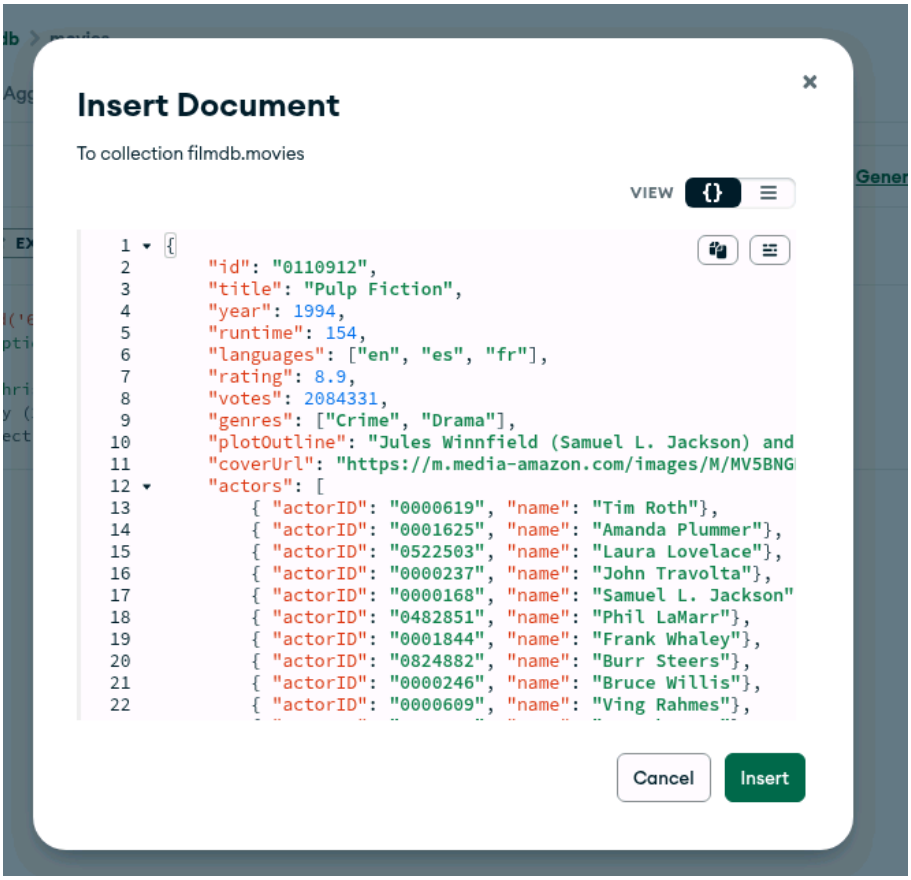
```
// Переключение на базу данных
use filmdb

// Проверка существующих коллекций
db.getCollectionNames()  // Вернёт: []

// Создание документа в новой коллекции
db.movies.insertOne({
    title: "Inception",
    year: 2010,
    director: "Christopher Nolan",
    genres: ["Sci-Fi", "Action"],
    ratings: {
        imdb: 8.8,
        metacritic: 74
    }
})
```

```
> use filmdb
switched to db filmdb
> db.getCollectionNames()
[ ]
> db.movies.insertOne({
...     title: "Inception",
...     year: 2010,
...     director: "Christopher Nolan",
...     genres: ["Sci-Fi", "Action"],
...     ratings: {
...         imdb: 8.8,
...         metacritic: 74
...     }
... })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("68bf5a3128f83874ea078bfc")
}
>
```

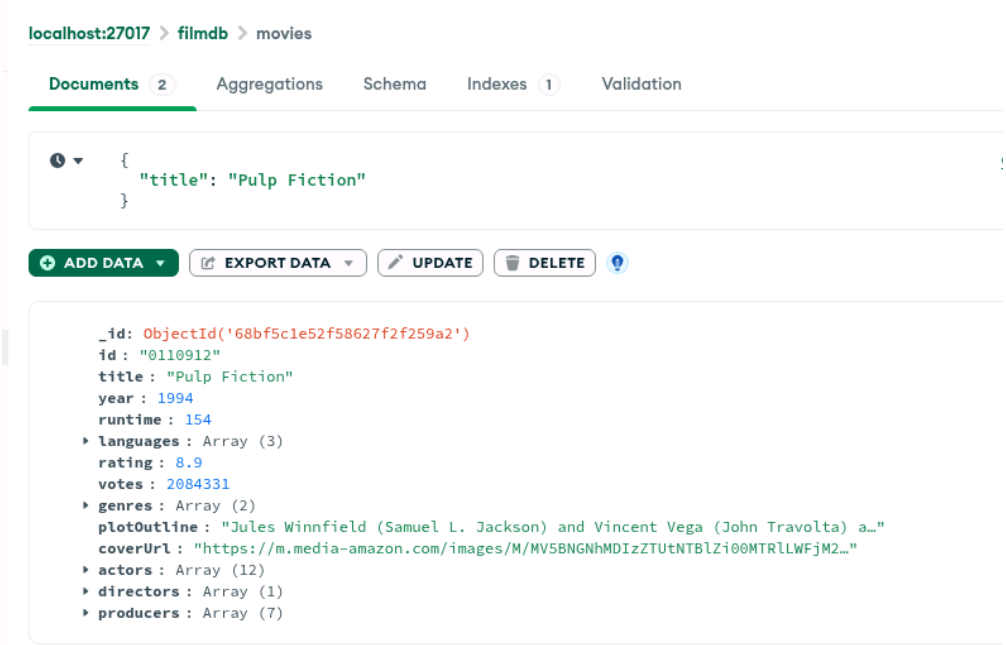
Практическая работа 1. Создание документов в MongoDB

Подключение и создание БД

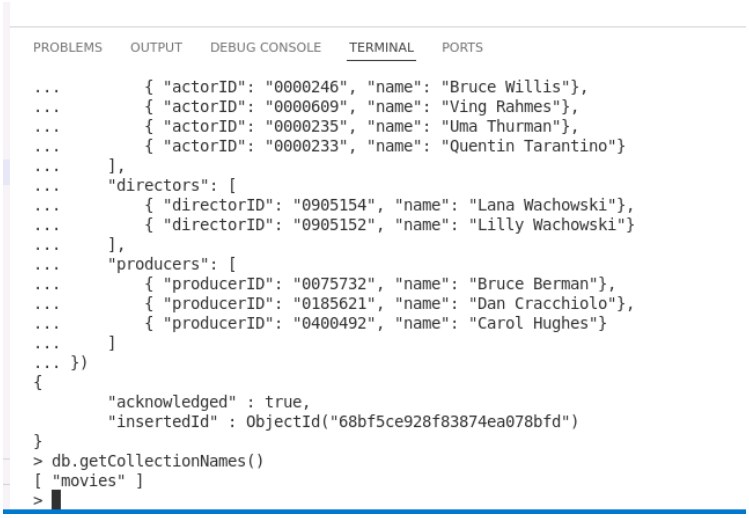




Поиск документа

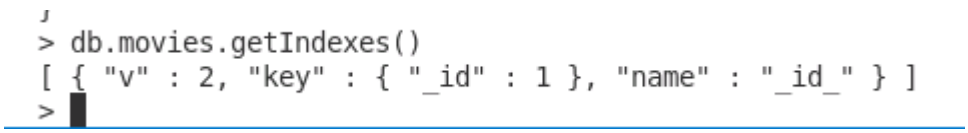


Вставка еще одного фильма и проверка



Вывод в отформатированном виде (список документов)

Поле id индексируется



(остановка контейнера sudo docker stop mongo-1)

Создание документов актеров в коллекции persons

```
    from pymongo import MongoClient

    # 1. Параметры подключения
    mongo_uri = "mongodb://root:abc123!@mongo-1:27017/"
    db_name = "student"
    collection_name = "test_labs"

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

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

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

        # 4. Вставка данных (Create)
        test_data = [
            {"lab_name": "Lab 1", "subject": "Physics",
"score": 85},
            {"lab_name": "Lab 2", "subject":
"Chemistry", "score": 90},
            {"lab_name": "Lab 3", "subject": "Biology",
"score": 88},
```

```
        ]
        result = collection.insert_many(test_data)
        print(f"\nВставлено документов:
{len(result.inserted_ids)}")

        # 5. Чтение данных (Read)
        print("\nСодержимое коллекции:")
        for doc in collection.find():
            print(doc)

        # 6. Обновление данных (Update)
        collection.update_one({"subject": "Physics"},
{"$set": {"score": 95}})
        print("\nДокумент после обновления:")
        print(collection.find_one({"subject":
"Physics"}))

        # 7. Удаление данных (Delete)
        collection.delete_one({"subject": "Chemistry"})
        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('68c927db3b2aae497ee74b87'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 85}
{'_id': ObjectId('68c927db3b2aae497ee74b88'), 'lab_name': 'Lab 2', 'subject': 'Chemistry', 'score': 90}
{'_id': ObjectId('68c927db3b2aae497ee74b89'), 'lab_name': 'Lab 3', 'subject': 'Biology', 'score': 88}

Документ после обновления:
{'_id': ObjectId('68c927db3b2aae497ee74b87'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'score': 95}

Количество документов после удаления: 2

Коллекция 'test_labs' удалена.

Подключение к MongoDB закрыто.
```

Задание Mongo

Найти все фильмы в жанре "Action", выпущенные после 2010 года, и увеличить ($inc) их счетчик голосов (votes) на 50.

```
mgpu@mgpu-vm:~/Downloads/idb/nosql-workshop/01-environment/docker$ sudo docker exec -it mongo-1 mongo -u root -p abc123! --authenticationDatabase admin
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("11d1d314-4a68-4e79-adf6-a1c19920602e") }
MongoDB server version: 4.4.29
---
The server generated these startup warnings when booting:
        2025-09-16T21:53:43.451+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodn
otes-filesystem
---
> use filmdb
switched to db filmdb
```

```
> db.movies.updateMany(
...   { genres: "Action", year: { $gte: 2011 } },
...   { $inc: { votes: 50 } }
... )
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
> db.movies.find(
...   { genres: "Action", year: { $gte: 2011 } },
...   { title: 1, year: 1, votes: 1, _id: 0 }
... ).sort({ year: 1 })
{ "title" : "John Wick", "year" : 2014, "votes" : 50 }
{ "title" : "Mad Max: Fury Road", "year" : 2015, "votes" : 50 }
{ "title" : "Avengers: Endgame", "year" : 2019, "votes" : 50 }
>
```

В Mongo Shell можно сделать “табличный” вывод с помощью метода forEach и форматирования строки

```
> db.movies.find(
...   { genres: "Action", year: { $gte: 2011 } },
...   { title: 1, year: 1, votes: 1, _id: 0 }
... ).sort({ year: 1 }).forEach(function(doc) {
...     print(doc.year + " | " + doc.title + " | Votes: " + doc.votes);
... })
2014 | John Wick | Votes: 50
2015 | Mad Max: Fury Road | Votes: 50
2019 | Avengers: Endgame | Votes: 50
```

Задание 2 (Neo4j)

Найти всех актеров, которые снимались в одних и тех же фильмах, что и "Tom Hanks" (коллеги по съемочной площадке).

```
MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(colleague:Person)
WHERE colleague.name <> "Tom Hanks"
RETURN DISTINCT colleague.name AS Actor, m.title AS Movie
ORDER BY colleague.name, m.title
```

MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED_IN]->(m:Movie)<-[:ACTED_IN]-(colleague:Person)

1. MATCH — основной оператор для поиска шаблонов в графе.
2. (tom:Person {name: "Tom Hanks"}) — ищем узел с меткой Person и свойством name = "Tom Hanks".
    - Этот узел мы называем tom.
3. -[:ACTED_IN]->(m:Movie) — идём по ребру ACTED_IN от Тома Хэнкса к фильмам.
    - m:Movie — узлы с меткой Movie.
4. <-[:ACTED_IN]-(colleague:Person) — идём от этих же фильмов обратно по ACTED_IN к другим актёрам.
    - Эти узлы называем colleague.

Итак, MATCH находит **все фильмы с Томом Хэнксом и всех актёров, кто там снимался**.

WHERE colleague.name <> "Tom Hanks"

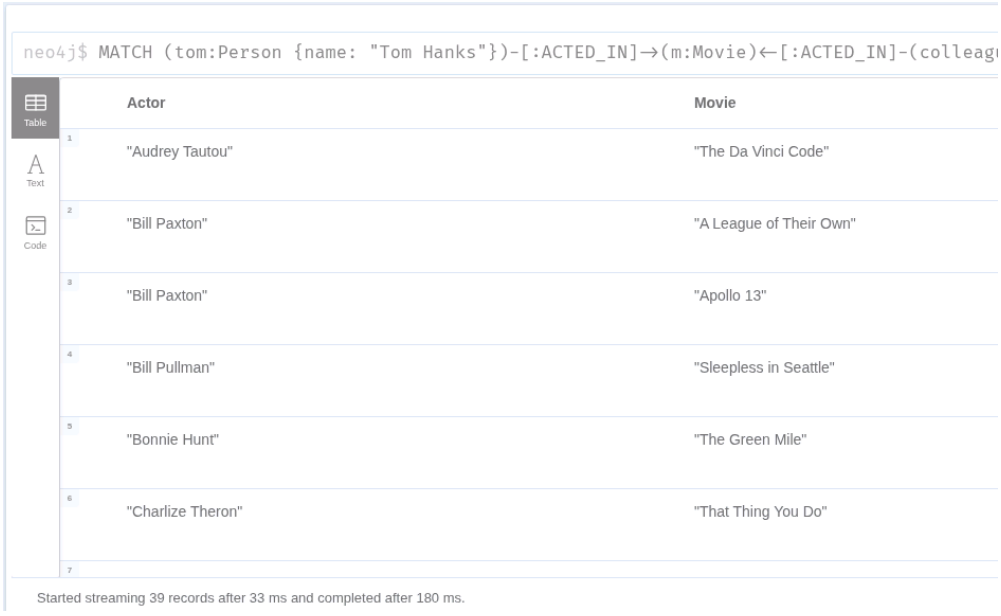
- Исключаем самого Тома Хэнкса из списка коллег.

RETURN DISTINCT colleague.name AS Actor, m.title AS Movie

1. RETURN — выбираем, что выводить в таблицу.
2. colleague.name AS Actor — имя актёра выводим под заголовком Actor.
3. m.title AS Movie — название фильма выводим под заголовком Movie.
4. DISTINCT — убирает дубликаты, если один актёр снимался с Томом в нескольких фильмах.

ORDER BY colleague.name, m.title

- Сортируем таблицу сначала по имени актёра, потом по названию фильма.



Задание 3 (Redis)

Смоделировать сессию пользователя: с помощью хэша (HSET) с ключом user:101 сохранить u sername и email. Установить срок жизни сессии в 300 секунд (EXPIRE).

```
redis-1:6379> HSET user:101 username "roni_voznes" email "roni@gmail.com"
(integer) 2
redis-1:6379> EXPIRE user:101 300
(integer) 1
redis-1:6379> HGETALL user:101
1) "username"
2) "roni_voznes"
3) "email"
4) "roni@gmail.com"
redis-1:6379> TTL user:101
(integer) 266
redis-1:6379>
```

HSET user:101 username "john_doe" email "john@example.com"

1. **HSET** — команда для работы с хэшами в Redis.
2. **user:101** — ключ хэша, т.е. имя объекта в Redis. Обычно используют user:<id>, чтобы легко находить пользователя.
3. **username "john_doe" email "john@example.com"** — поля и их значения. В хэше можно хранить много пар field → value.
4. **Что делает:** создаёт хэш user:101 с двумя полями:
    - username = "john_doe"
    - email = "john@example.com"

EXPIRE user:101 300

1. **EXPIRE** — команда для установки времени жизни ключа в секундах.
2. **user:101** — ключ, на который ставим TTL (time to live).
3. **300** — время жизни в секундах (5 минут).
4. **Что делает:** через 300 секунд Redis автоматически удалит ключ user:101 вместе с хэшем.

Итого:

- Ты создал «сессию» пользователя в виде хэша.
- Хэш содержит поля username и email.

- Сессия живёт 5 минут, после чего Redis её автоматически удаляет.

Если хочешь, могу нарисовать маленькую **схему**, как это хранится в Redis и как работает TTL. Это сильно помогает визуально понять.

Хочешь такую схему?