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

A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Запуск sudo docker compose up -d

sudo docker compose stop

(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

A screenshot of a computer

AI-generated content may be incorrect.

http://localhost:28203/

http://localhost:28204/A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with a message

AI-generated content may be incorrect.

Добавили подключение

A screenshot of a computer code

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

**Практическая работа 1. Создание документов в MongoDB**

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Подключение и создание БД

A screenshot of a computer

AI-generated content may be incorrect.Поиск документа

A computer screen shot of a computer code

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.Вставка еще одного фильма и проверка

Вывод в отформатированном виде (список документов)

Поле id индексируется  
A close up of a text

AI-generated content may be incorrect.

(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}")

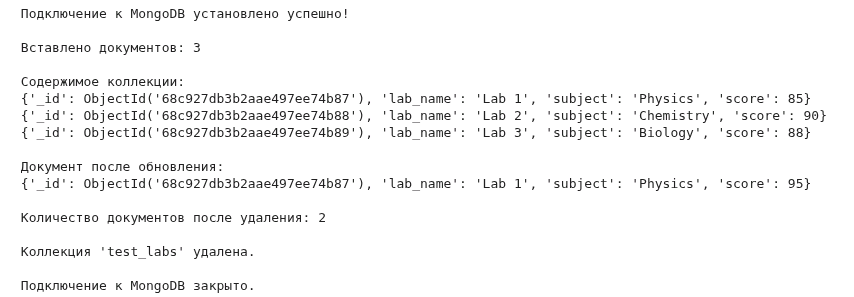
finally:

# 9. Закрытие подключения

if 'client' in locals() and client:

client.close()

print("\nПодключение к MongoDB закрыто.")



Задание Mongo

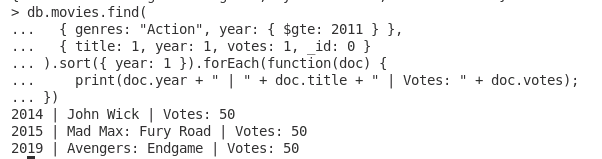
A close up of a text

AI-generated content may be incorrect.Найти все фильмы в жанре "Action", выпущенные после 2010 года, и увеличить ($inc) их счетчик голосов (votes) на 50.

A screenshot of a computer code

AI-generated content may be incorrect.

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



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

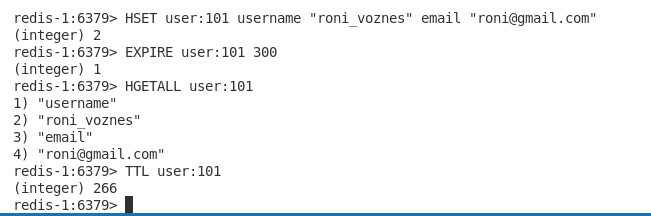
* Сортируем таблицу сначала по имени актёра, потом по названию фильма.

A screenshot of a computer

AI-generated content may be incorrect.

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).



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**](mailto: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?