Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Инструменты для хранения и обработки больших данных

Лабораторная работа №2.1

Тема:

«Изучение методов хранения данных на основе NoSQL.**»**

Выполнила: Ванярина Ю. А., группа: АДЭУ-211

Преподаватель: Босенко Т. М.

Москва

2024

Теоретическая часть

MongoDB: документо-ориентированная NoSQL база данных, где данные хранятся в формате JSON-подобных документов.

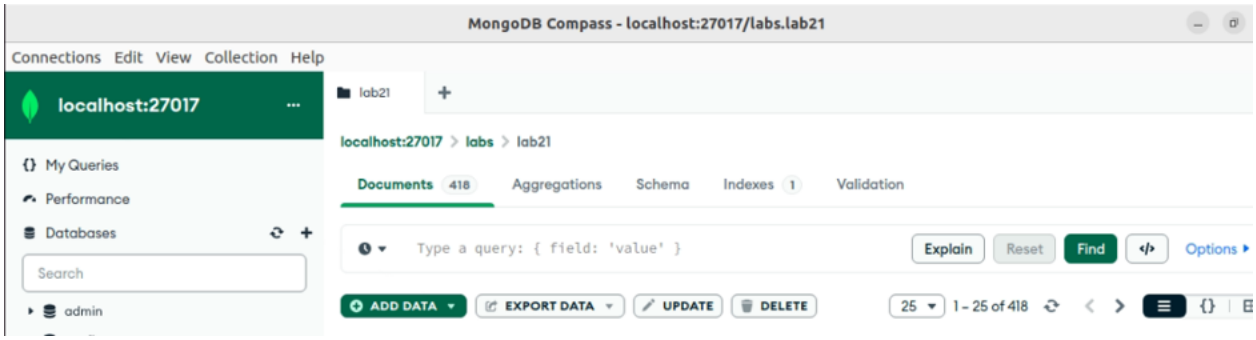
Redis: высокопроизводительная база данных типа "ключ-значение", часто используемая для кеширования и временного хранения данных.

Neo4j: графовая база данных, которая позволяет хранить данные в виде вершин и рёбер графа, что удобно для моделирования сложных взаимосвязей.

Ход работы

Проверить соединение с Mongo

Запустить Mongo DB Compass



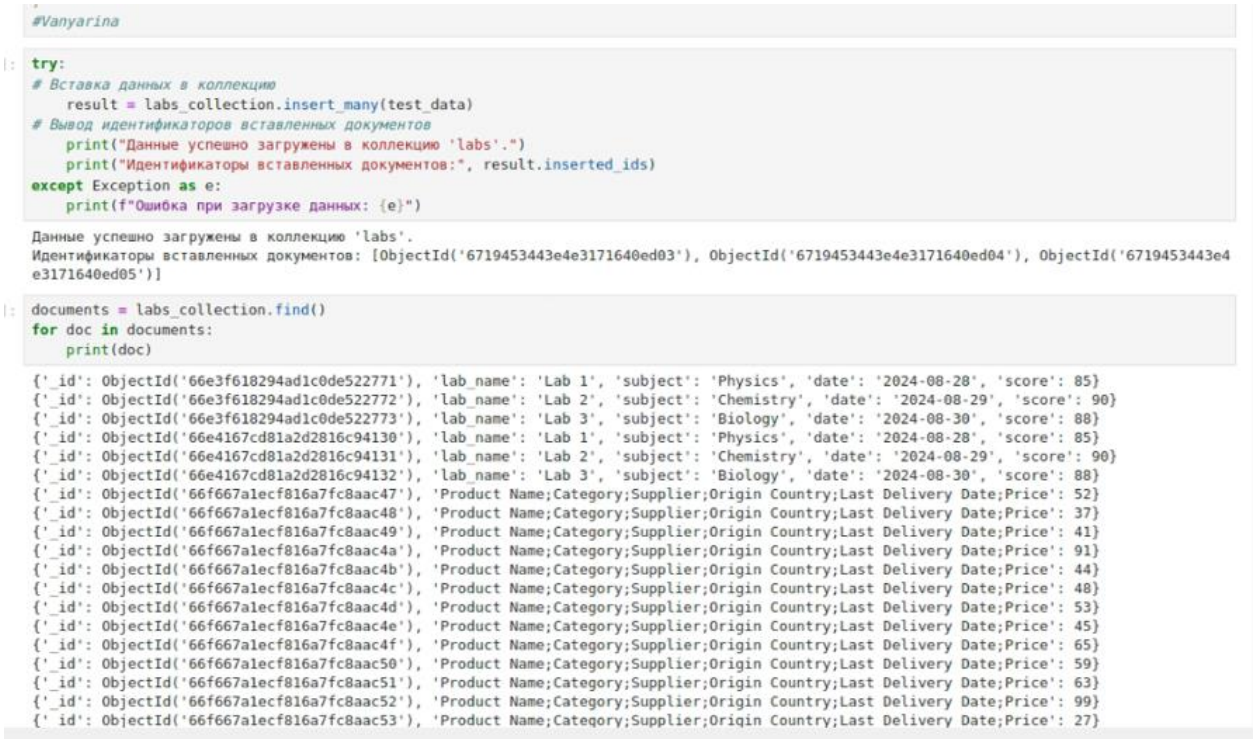
Шаг 1: Установка и настройка MongoDB в Jupyter Hub

Установка необходимых библиотек

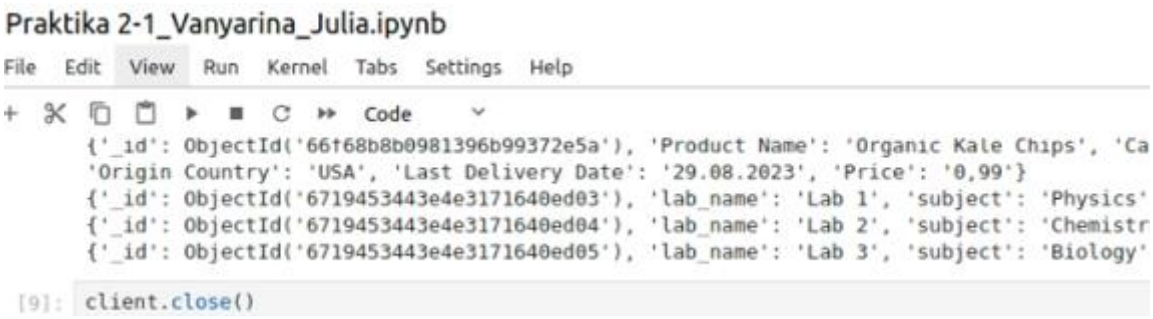
Подключение к MongoDB с аутентификацией

Создание тестовых данных

Загрузка данных в коллекцию labs

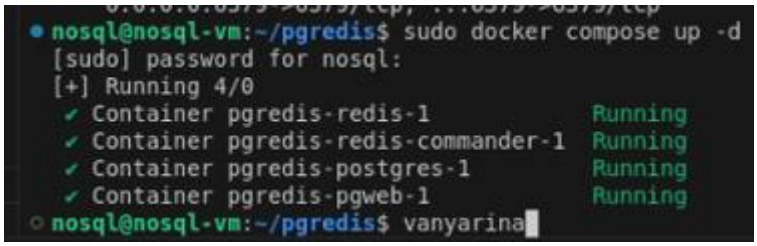


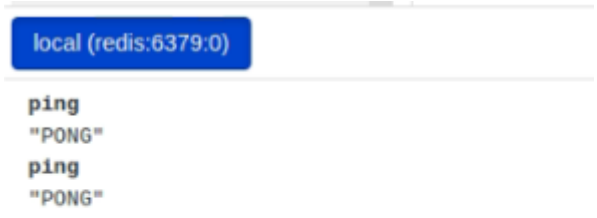
Закрытие подключения



Этот пример демонстрирует, как подключиться к MongoDB с использованием аутентификации, выбрать базу данных и коллекцию, выполнить операции с данными и закрыть соединение.

Проверить соединение с Redis





2.4 После установки библиотеки можно подключиться к Redis и выполнить необходимые операции

```python
# Подключение к Redis с аутентификацией
r = redis.Redis(
host='localhost',
port=6379,
db=0 # Подключение к базе данных 0
)
# Проверка соединения
try:
    r.ping()
    print("Соединение с Redis успешно установлено.")
except redis.ConnectionError:
    print("Не удалось подключиться к Redis.")
#Vanyarina
```

```
Соединение с Redis успешно установлено.
```

Генерация и загрузка данных в Redis. Сгенерируем десять записей и загрузим их в базу данных Redis.

```python
from datetime import datetime
# Создание 10 записей
for i in range(1, 11):
    key = f"key_{i}"
    value = f"value_{i}"
    r.set(key, value)
    print(f"Создана запись: {key} = {value}")
# Проверка созданных записей
for i in range(1, 11):
    key = f"key_{i}"
    value = r.get(key)
    print(f"Проверка: {key} = {value.decode('utf-8')}")
#Vanyarina
```

```
Создана запись: key_1 = value_1
Создана запись: key_2 = value_2
Создана запись: key_3 = value_3
Создана запись: key_4 = value_4
Создана запись: key_5 = value_5
Создана запись: key_6 = value_6
Создана запись: key_7 = value_7
Создана запись: key_8 = value_8
Создана запись: key_9 = value_9
Создана запись: key_10 = value_10
Проверка: key_1 = value_1
Проверка: key_2 = value_2
Проверка: key_3 = value_3
Проверка: key_4 = value_4
Проверка: key_5 = value_5
Проверка: key_6 = value_6
Проверка: key_7 = value_7
Проверка: key_8 = value_8
Проверка: key_9 = value_9
Проверка: key_10 = value_10
```

```python
#Функция для печати разделителя
def print_separator(message):
    print(f"\n{'='*20} {message} {'='*20}")
#Vanyarina
```

Создание по 5 записей различных типов данных.

[17]:
```
print_separator("Создание данных")
# Строки
for i in range(1, 6):
    r.set(f"string_{i}", f"value_{i}")
    print(f"Создана строка: string_{i} = value_{i}")

# Списки
for i in range(1, 6):
    r.rpush(f"list_{i}", *[f"item_{j}" for j in range(1, 4)])
    print(f"Создан список: list_{i} = {r.lrange(f'list_{i}', 0, -1)}")

# Множества
for i in range(1, 6):
    r.sadd(f"set_{i}", *[f"element_{j}" for j in range(1, 4)])
    print(f"Создано множество: set_{i} = {r.smembers(f'set_{i}')}")

# Хэши
for i in range(1, 6):
    r.hset(f"hash_{i}", mapping={f"field_{j}": f"value_{j}" for j in range(1, 4)})
    print(f"Создан хэш: hash_{i} = {r.hgetall(f'hash_{i}')}")

# Упорядоченные множества
for i in range(1, 6):
    r.zadd(f"zset_{i}", {f"member_{j}": j for j in range(1, 4)})
    print(f"Создано упорядоченное множество: zset_{i} = {r.zrange(f'zset_{i}', 0, -1, withscores=True)}")
#Vanyarina
```

```
========== Создание данных ==========
Создана строка: string_1 = value_1
Создана строка: string_2 = value_2
```

Получение данных по ключу.

```
print_separator("Получение данных")
print(f"Строка: {r.get('string_1')}")
print(f"Список: {r.lrange('list_1', 0, -1)}")
print(f"Множество: {r.smembers('set_1')}")
print(f"Хэш: {r.hgetall('hash_1')}")
print(f"Упорядоченное множество: {r.zrange('zset_1', 0, -1, withscores=True)}")
#Vanyarina
```

```
========== Получение данных ==========
Строка: b'value_1'
Список: [b'item_1', b'item_2', b'item_3']
Множество: {b'element_1', b'element_3', b'element_2'}
Хэш: {b'field_1': b'value_1', b'field_2': b'value_2', b'field_3': b'value_3'}
Упорядоченное множество: [(b'member_1', 1.0), (b'member_2', 2.0), (b'member_3', 3.0)]
```

2.9Удаления записей

```
========== Обновление данных ==========
Обновлена строка: string_1 = b'new_value'
Обновлен список: list_1 = [b'new_item', b'item_2', b'item_3']
Обновлено множество: set_1 = {b'element_1', b'element_3', b'element_2', b'new_element'}
Обновлен хэш: hash_1 = {b'field_1': b'value_1', b'field_2': b'value_2', b'field_3': b'value_3', b'new_field': b'new_value'}
Обновлено упорядоченное множество: zset_1 = [(b'member_1', 1.0), (b'member_2', 2.0), (b'member_3', 3.0), (b'new_member', 5.0)]
```

[20]:
```
print_separator("Удаление данных")
r.delete("string_5", "list_5", "set_5", "hash_5", "zset_5")
print("Удалены ключи: string_5, list_5, set_5, hash_5, zset_5")
#Vanyarina
```

```
========== Удаление данных ==========
Удалены ключи: string_5, list_5, set_5, hash_5, zset_5
```

[21]:
```
print_separator("Проверка удаленных данных")
for key in ["string_5", "list_5", "set_5", "hash_5", "zset_5"]:
    print(f"Существует ли ключ {key}? {r.exists(key)}")
#Vanyarina
```

```
========== Проверка удаленных данных ==========
Существует ли ключ string_5? 0
Существует ли ключ list_5? 0
Существует ли ключ set_5? 0
Существует ли ключ hash_5? 0
Существует ли ключ zset 5? 0
```

Выгрузить все данные из Redis в csv.

[23]:
```
def flatten_data(data):
    if isinstance(data, (str, int, float)):
        return str(data)
    elif isinstance(data, list):
        return json.dumps(data, ensure_ascii=False)
    elif isinstance(data, dict):
        return json.dumps(data, ensure_ascii=False)
    else:
        return str(data)
#Vanyarina
```

*[20]:
```
def dump_redis_to_csv(filename='redis_dump.csv'):
```

```
r = redis.Redis(host='localhost', port=6379, db=0)

# Получение всех ключей
keys = r.keys('*')
with open(filename, 'w', newline='', encoding='utf-8') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerow(['Key', 'Type', 'Value']) # Заголовки
    for key in keys:

        # Декодирование ключа из байтов в строку
        key_str = key.decode('utf-8')

        # Определение типа данных ключа
        key_type = r.type(key).decode('utf-8')
        if key_type == 'string':
            value = r.get(key).decode('utf-8')
        elif key_type == 'list':
            value = r.lrange(key, 0, -1)
            value = [item.decode('utf-8') for item in value]
        elif key_type == 'set':
            value = list(r.smembers(key))
            value = [item.decode('utf-8') for item in value]
        elif key_type == 'hash':
            value = r.hgetall(key)
            value = {k.decode('utf-8'): v.decode('utf-8') for k, v in value.items()}
        elif key_type == 'zset':
            value = r.zrange(key, 0, -1, withscores=True)
            value = [(item[0].decode('utf-8'), item[1]) for item in value]
        else:
            value = f"Неподдерживаемый тип данных: {key_type}"
# Записываем данные в CSV
        csvwriter.writerow([key_str, key_type, flatten_data(value)])
# Закрытие соединения
r.close()
print(f"Данные сохранены в файл '{filename}'")
```

```
Данные сохранены в файл 'redis_dump.csv'

[27]: ls

  Desktop/                          Templates/
  Documents/                        Untitled.ipynb
  Downloads/                        Videos/
  MongoDB-Copy1.ipynb               google-chrome-stable_current_amd64.deb
  MongoDB.ipynb                     mongoDB.ipynb
  Music/                            mongodb/
  Pictures/                         pgredis/
 'Praktika 2-1_Vanyarina_Julia.ipynb'  redis_dump.csv
  Public/                           snap/

[ ]:
```

Neo4j

Neo4j — это графовая система управления базами данных с открытым исходным кодом, реализованная на Java.

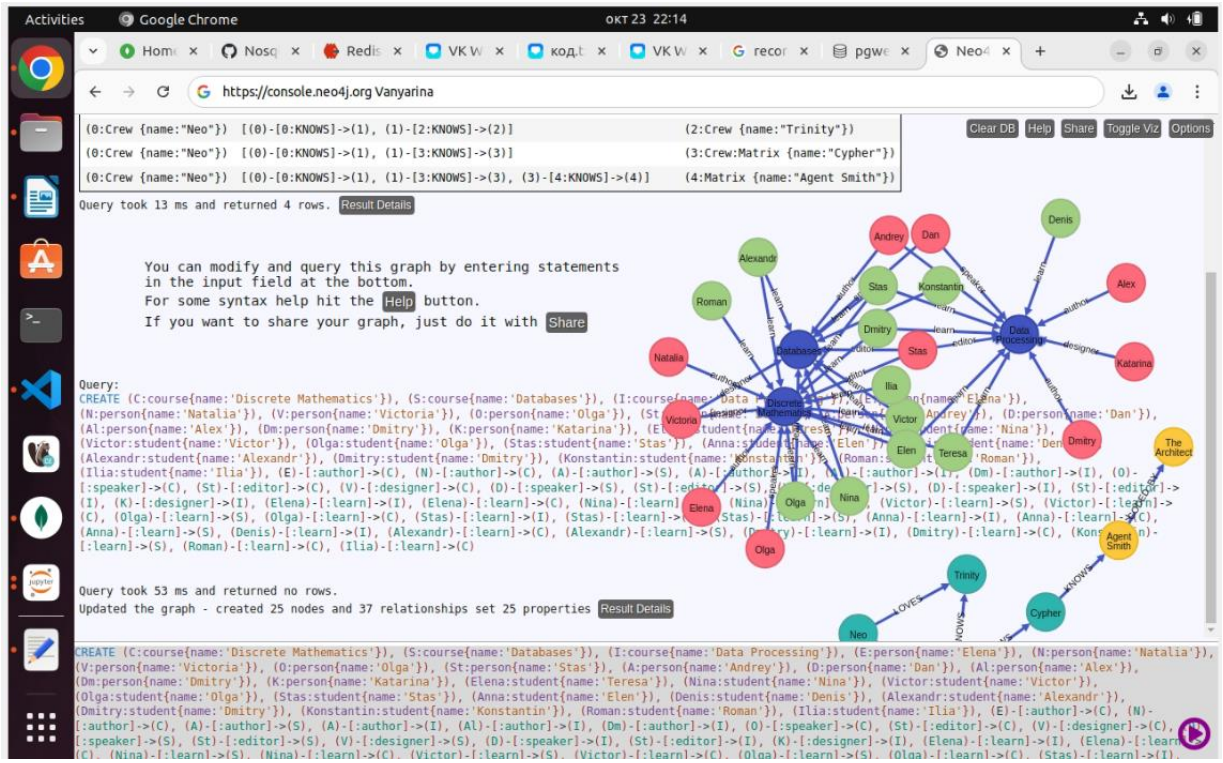
Модель Neo4j в основном состоит из следующих основных компонентов:

— Узлы (эквивалентны вершинам в теории графов): основные элементы данных, которые связаны отношениями. Узлы могут иметь метки и свойства.

— Отношения (эквивалентно ребрам в теории графов): описывает соединения между узлами и соединяет их вместе. Отношения могут иметь одно или несколько свойств.

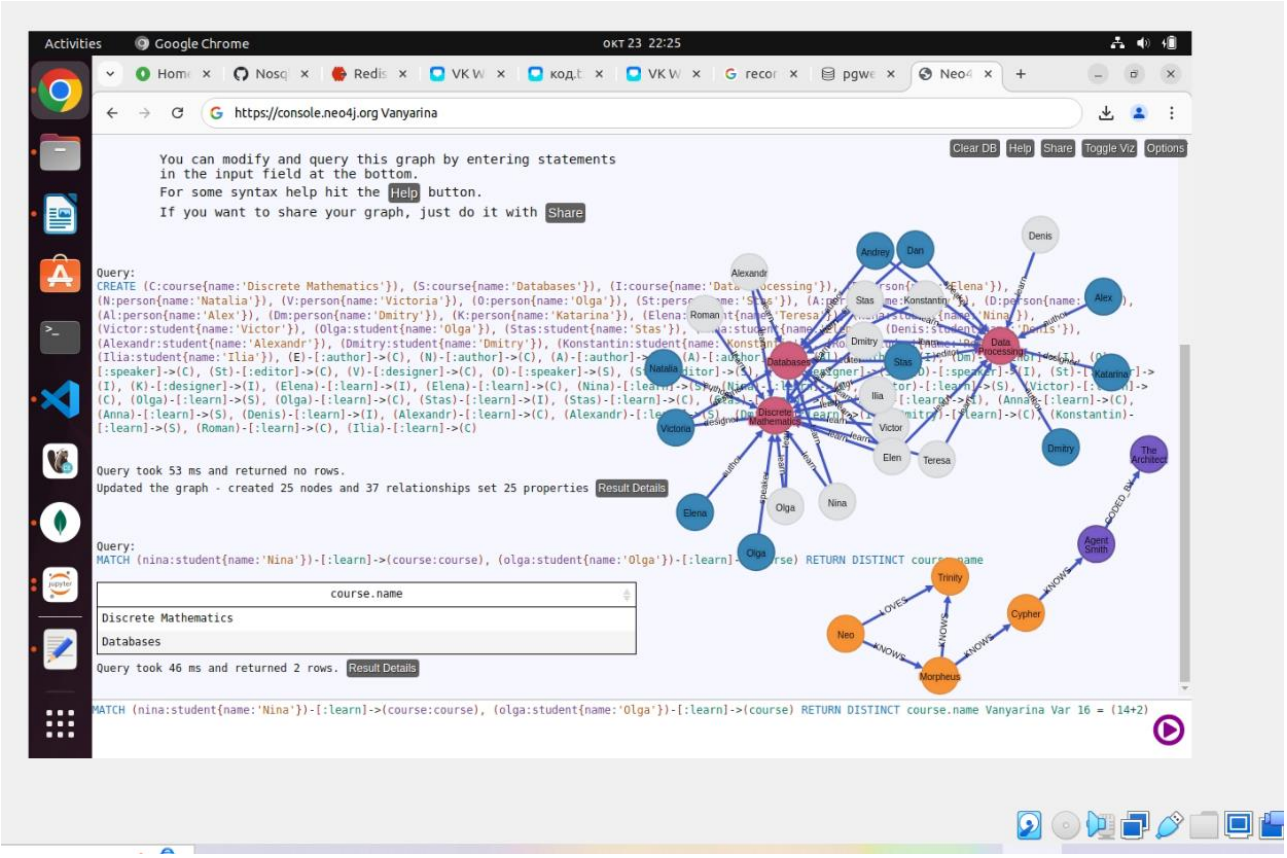
— Метки: представляют роль узлов. Метки используются для группировки узлов. У каждого узла может быть несколько меток. Метки также индексируются для ускорения процесса поиска узлов в графе.

— Свойства: атрибуты узлов и отношений, включающих пары имен или значений.



Вариант 2. Self: Составьте запрос, который вернет список курсов, на которые записаны студенты с именем "Nina" и "Olga"

MATCH(nina:student{name:'Nina'})-[:learn]->(course:course), (olga:student{name:'Olga'})-[:learn]->(course) RETURN DISTINCT course.name



Индивидуальное задание.Вариант 16.

Оборудование и его спецификации в производстве:

— ID оборудования

— Название оборудования

— Тип

— Производитель

— Дата установки

— Мощность

— Габариты (ДхШхВ)

— Вес (кг)

— Срок службы (лет)

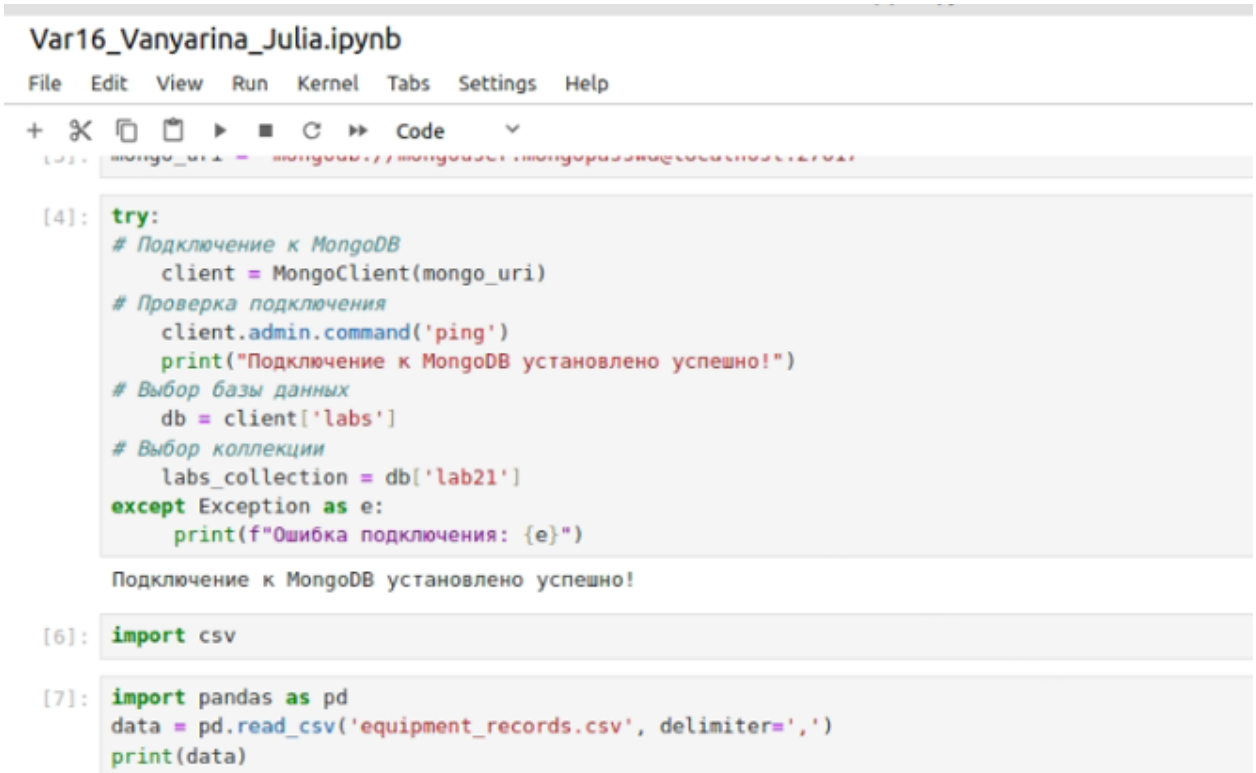
— Статус (работает/на обслуживании)

Проверяем соединение в Монго и импортируем данные

```

*ID,name,type,manufactor,date_instalation,power,size,weight,service_life,status 1,Robotic arm,Mechanical,Haas,2013-11-04,1168,4455x2860x1144,8725,24,works 2,Milling machine,Laser,Mazak,2024-08-15,4946,4541x1194x1151,9434,16,in service 3,Grinding machine,Laser,Yaskawa,2007-12-14,2759,1085x2733x2035,6832,25,works 4.3D Printer,Robotic,Fanuc,2012-12-05,3036,2291x1253x1203,2424,15,works 5,Milling machine,Mechanical,ABB,2017-08-23,3389,3213x2321x895,6384,11,works 6.3D printer,Laser,Siemens,2013-10-08,4228,2650x2618x2033,682,14,works*

```



Создание новой коллекции

```
[18]: new_collection.insert_many(df)

print(f"Коллекция '{collection_name}' успешно создана и данные добавлены.")
```

```
Коллекция 'new_collection' успешно создана и данные добавлены.
```

```
[19]: print(df)
```


```
[{'equipment_id': 1, 'equipment_name': 'Milling Machine', 'type': 'Pneumatic', 'manufacturer': 'Siemens', 'installation_date': '2021-07-3
0', 'power': 2110, 'dimensions': '2113x913x2950', 'weight': 9367, 'service_life_years': 20, 'status': 'operational', '_id': ObjectId('6718
16d48d0011768ff5355d')}, {'equipment_id': 2, 'equipment_name': 'Plasma Cutting Machine', 'type': 'Laser', 'manufacturer': 'Haas', 'install
ation_date': '2008-05-29', 'power': 3824, 'dimensions': '3501x1443x1102', 'weight': 7206, 'service_life_years': 11, 'status': 'under maint
enance', '_id': ObjectId('671816d48d0011768ff5355e')}, {'equipment_id': 3, 'equipment_name': 'Laser Cutter', 'type': 'Hydraulic', 'manufac
turer': 'Mitsubishi', 'installation_date': '2005-01-06', 'power': 965, 'dimensions': '2099x992x1199', 'weight': 3880, 'service_life_year
s': 9, 'status': 'operational', '_id': ObjectId('671816d48d0011768ff5355f')}, {'equipment_id': 4, 'equipment_name': 'Hydraulic Press', 'ty
pe': 'Electrical', 'manufacturer': 'Haas', 'installation_date': '2018-01-03', 'power': 273, 'dimensions': '3653x1443x2600', 'weight': 603
2, 'service_life_years': 16, 'status': 'operational', '_id': ObjectId('671816d48d0011768ff53560')}, {'equipment_id': 5, 'equipment_name':
'Grinding Machine', 'type': 'Mechanical', 'manufacturer': 'Haas', 'installation_date': '2007-08-16', 'power': 4764, 'dimensions': '1009x27
82x2616', 'weight': 7725, 'service_life_years': 13, 'status': 'operational', '_id': ObjectId('671816d48d0011768ff53561')}, {'equipment_i
d': 6, 'equipment_name': 'Metal Press', 'type': 'Pneumatic', 'manufacturer': 'ABB', 'installation_date': '2006-11-12', 'power': 103, 'dime
nsions': '2266x2821x1596', 'weight': 429, 'service_life_years': 22, 'status': 'operational', '_id': ObjectId('671816d48d0011768ff53562')},
{'equipment_id': 7, 'equipment_name': 'Hydraulic Press', 'type': 'Electrical', 'manufacturer': 'Fanuc', 'installation_date': '2014-11-22',
'power': 4699, 'dimensions': '4713x1165x742', 'weight': 6081, 'service_life_years': 20, 'status': 'operational', '_id': ObjectId('671816d4
8d0011768ff53563')}, {'equipment_id': 8, 'equipment_name': 'Hydraulic Press', 'type': 'Laser', 'manufacturer': 'Fanuc', 'installation_dat
e': '2016-08-01', 'power': 2135, 'dimensions': '2610x2886x1087', 'weight': 9516, 'service_life_years': 13, 'status': 'operational', '_id':
```


Поиск записей с наименованием гидравлический пресс

```
20]: documents = new_collection.find({'equipment_name': 'Hydraulic Press'})
for doc in documents:
    print(doc)
```


```
{'_id': ObjectId('671816d48d0011768ff53560'), 'equipment_id': 4, 'equipment_name': 'Hydraulic Press', 'type': 'Electrical', 'manufacture
r': 'Haas', 'installation_date': '2018-01-03', 'power': 273, 'dimensions': '3653x1443x2600', 'weight': 6032, 'service_life_years': 16, 'st
atus': 'operational'}
{'_id': ObjectId('671816d48d0011768ff53563'), 'equipment_id': 7, 'equipment_name': 'Hydraulic Press', 'type': 'Electrical', 'manufacture
r': 'Fanuc', 'installation_date': '2014-11-22', 'power': 4699, 'dimensions': '4713x1165x742', 'weight': 6081, 'service_life_years': 20, 's
tatus': 'operational'}
{'_id': ObjectId('671816d48d0011768ff53564'), 'equipment_id': 8, 'equipment_name': 'Hydraulic Press', 'type': 'Laser', 'manufacturer': 'Fa
nuc', 'installation_date': '2016-08-01', 'power': 2135, 'dimensions': '2610x2886x1087', 'weight': 9516, 'service_life_years': 13, 'statu
s': 'operational'}
{'_id': ObjectId('671816d48d0011768ff53565'), 'equipment_id': 9, 'equipment_name': 'Hydraulic Press', 'type': 'Mechanical', 'manufacture
r': 'Mitsubishi', 'installation_date': '2007-02-07', 'power': 3050, 'dimensions': '3048x984x1418', 'weight': 2658, 'service_life_years': 1
5, 'status': 'operational'}
```


Выборка по типу оборудования Электрический

```
]: documents = new_collection.find({'type': 'Electrical'})
for doc in documents:
    print(doc)
```


```
{'_id': ObjectId('671816d48d0011768ff53560'), 'equipment_id': 4, 'equipment_name': 'Hydraulic Press', 'type': 'Electrical', 'manufacture
r': 'Haas', 'installation_date': '2018-01-03', 'power': 273, 'dimensions': '3653x1443x2600', 'weight': 6032, 'service_life_years': 16, 'st
atus': 'operational'}
{'_id': ObjectId('671816d48d0011768ff53563'), 'equipment_id': 7, 'equipment_name': 'Hydraulic Press', 'type': 'Electrical', 'manufacture
r': 'Fanuc', 'installation_date': '2014-11-22', 'power': 4699, 'dimensions': '4713x1165x742', 'weight': 6081, 'service_life_years': 20, 's
tatus': 'operational'}
```


Удаление производителя Haas

```
[24]: db.new_collection.delete_many({'manufacturer': 'Haas'})
```

```
[24]: DeleteResult({'n': 4, 'ok': 1.0}, acknowledged=True)
```

```
[ ]: #Vanyarina
```

Подключение к Redis

[26]:
```
import redis
# Подключение к Redis с аутентификацией
r = redis.Redis(
host='localhost',
port=6379,
db=0 # Подключение к базе данных 0
)
# Проверка соединения
try:
    r.ping()
    print("Соединение с Redis успешно установлено.")
except redis.ConnectionError:
    print("Не удалось подключиться к Redis.")
```

```
Соединение с Redis успешно установлено.
```

:
```
# Перебор строк и добавление их в Redis
for index, row in data.iterrows():
    # Преобразуем строку в словарь
    item_dict = row.to_dict()
```

:
```
r.hmset(f'record:{index}', item_dict)
```

```
/tmp/ipykernel_40872/3935466372.py:1: DeprecationWarning: Redis.hmset() is deprecated. Use Redis.hset() instead.
  r.hmset(f'record:{index}', item_dict)
```

:
```
True
```

:
```
def print_separator(message):
    print(f"\n{'='*20} {message} {'='*20}")
```

[42]:
```
r.set("1", "Robot")

print("Данные успешно загружены в Redis.")
```

```
Данные успешно загружены в Redis.
```

[43]:
```
#Используем уникальный ключ для каждой записи, например, id
a = r.get(1)
print(a)
```

```
b'Robot'
```

[44]:
```
print_separator("Обновление данных")
r.set("17", "Robot ultra 2000")
```

```
==================== Обновление данных ====================
```

[44]:
```
True
```

[45]:
```
a = r.get(1)
print(a)
```

```
b'Robot'
```

[46]:
```
r.delete(1)
```

[46]:
```
1
```

```
[46]: r.delete(1)

[46]: 1

[48]: print_separator("Проверка удаленных данных")
      a = r.get(1)
      print(a)

      ==================== Проверка удаленных данных ====================
      None
```

Выводы по использованию инструментов по хранению данных nosql

1. MongoDB:

MongoDB является документо-ориентированной NoSQL базой данных, которая хранит данные в формате документов JSON.

+MongoDB хорошо подходит для хранения больших объемов текстовых данных.

-Было сложно разобраться с установкой монго,так как последняя версия не подходила, были перепробованы все версии до 2016 года. Много раз не запускался контейнер.При подключении в монго компас отсутствовала бд student.Получилось выполнить задание после 8 попыток установки на разные конфигурации вм.

2. Redis:

Redis - это высокопроизводительная база данных типа ключ-значение, которая часто используется для сеансов хранения кеширования и обработки очередей сообщений.

+Его скорость и способность к хранению данных в оперативной памяти делают его отличным выбором для обработки высоконагруженных приложений.

3. Neo4j:

Neo4j - графовая база данных, которая хранит данные в виде узлов, связей и свойств. Он отлично подходит для хранения и обработки сложных структур данных, таких как социальные сети.

+Neo4j обеспечивает эффективную работу с запросами, а также обеспечивает глубокие аналитические возможности.

Понятнее и удобнее всего работа проходила с Redis.