

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:

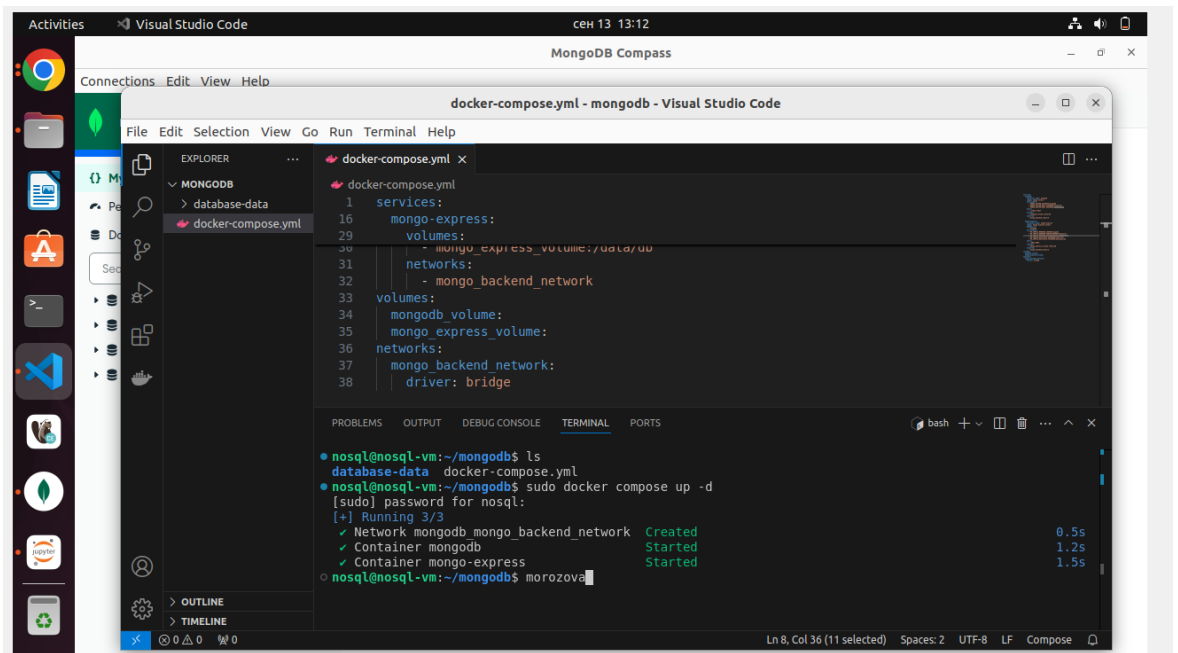
«Изучение методов хранения данных на основе NoSQL»

Выполнил(а): Морозова Валерия АДЭУ-211

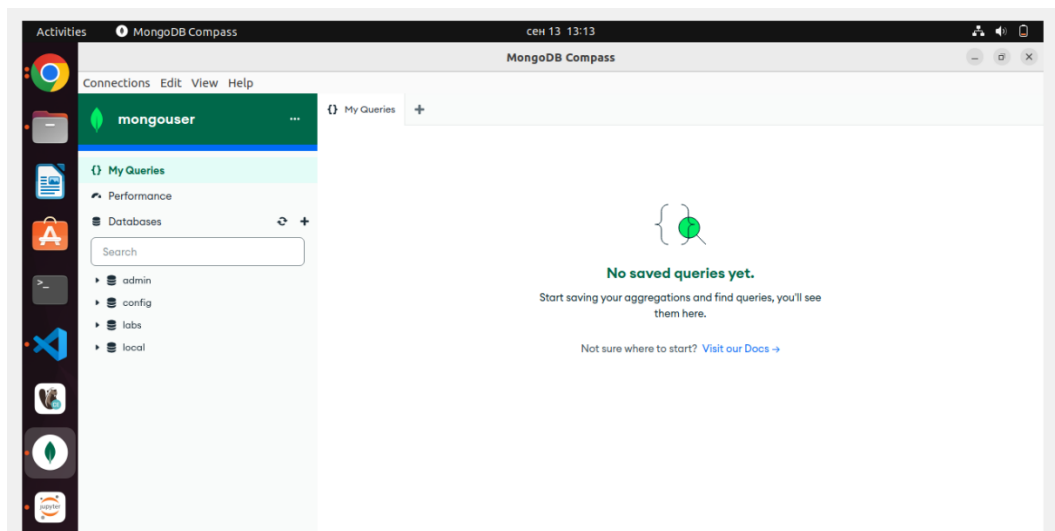
Преподаватель: Босенко Т.М.

Москва

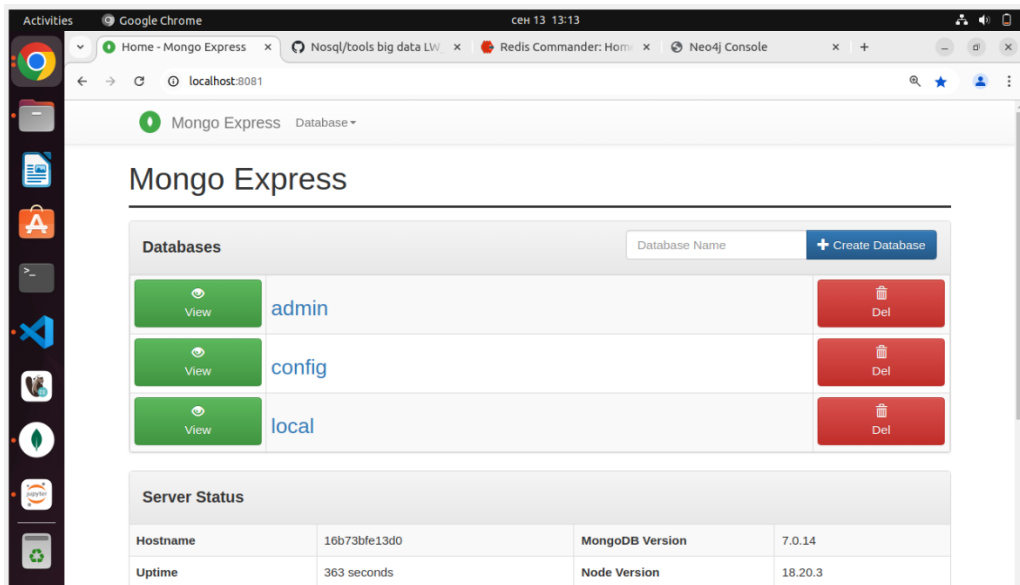
2024



1. Подключение к MongoDB с аутентификацией

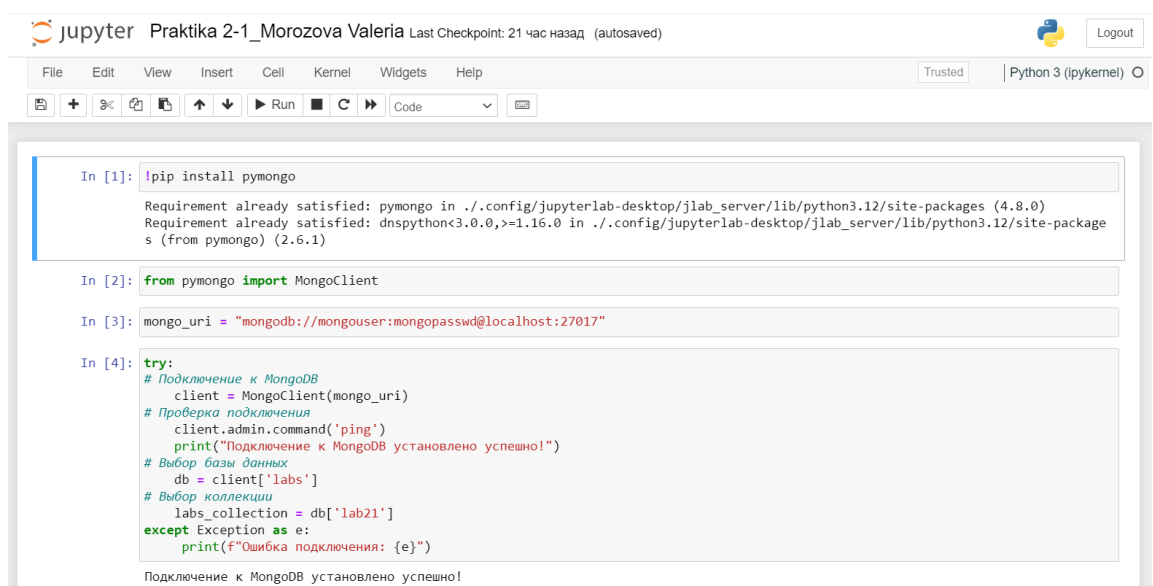


2. Открытие баз данных



3. Удалили лишние DB

Выполнение общей практической части:



4. Подключение к MongoDB в Jupyter

jupyter Praktika 2-1_Morozova Valeria Last Checkpoint: 21 час назад (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) C

```
In [5]: test_data = [
        {"lab_name": "Lab 1", "subject": "Physics", "date": "2024-08-28", "score": 85},
        {"lab_name": "Lab 2", "subject": "Chemistry", "date": "2024-08-29", "score": 90},
        {"lab_name": "Lab 3", "subject": "Biology", "date": "2024-08-30", "score": 88},
        ]

In [6]: try:
        # Вставка данных в коллекцию
        result = labs_collection.insert_many(test_data)
        # Вывод идентификаторов вставленных документов
        print("Данные успешно загружены в коллекцию 'labs'.")
        print("Идентификаторы вставленных документов:", result.inserted_ids)
    except Exception as e:
        print(f"Ошибка при загрузке данных: {e}")

Данные успешно загружены в коллекцию 'labs'.
Идентификаторы вставленных документов: [ObjectId('66e3f618294ad1c0de522771'), ObjectId('66e3f618294ad1c0de522772'), ObjectId('66e3f618294ad1c0de522773')]
```

5. Написание и вставка данных в коллекцию

jupyter Praktika 2-1_Morozova Valeria Last Checkpoint: 21 час назад (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) C

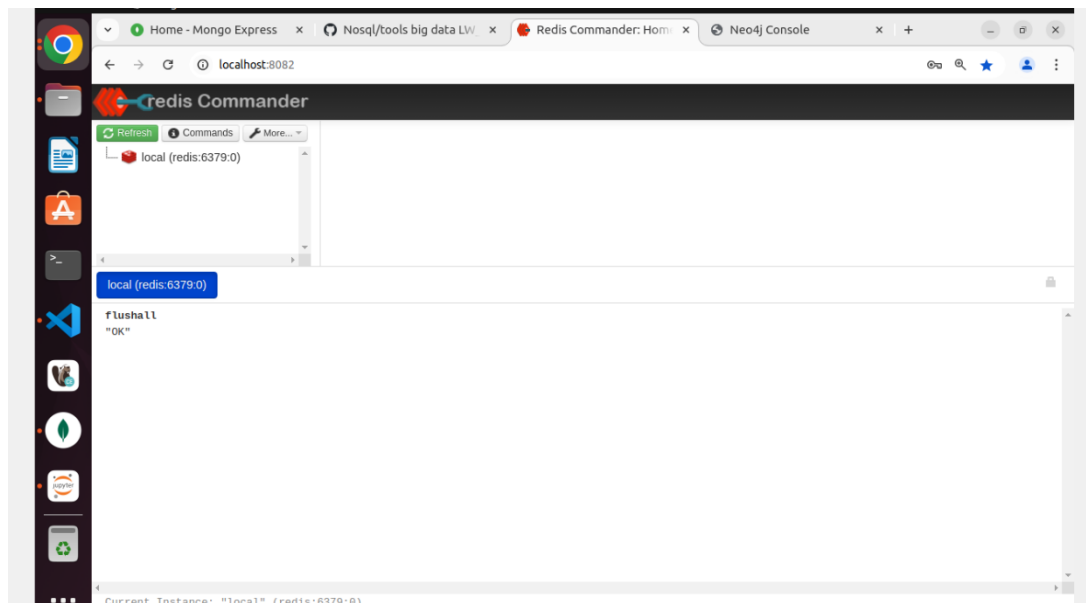
```
Идентификаторы вставленных документов: [ObjectId('66e3f618294ad1c0de522771'), ObjectId('66e3f618294ad1c0de522772'), ObjectId('66e3f618294ad1c0de522773')]

In [7]: documents = labs_collection.find()
        for doc in documents:
            print(doc)

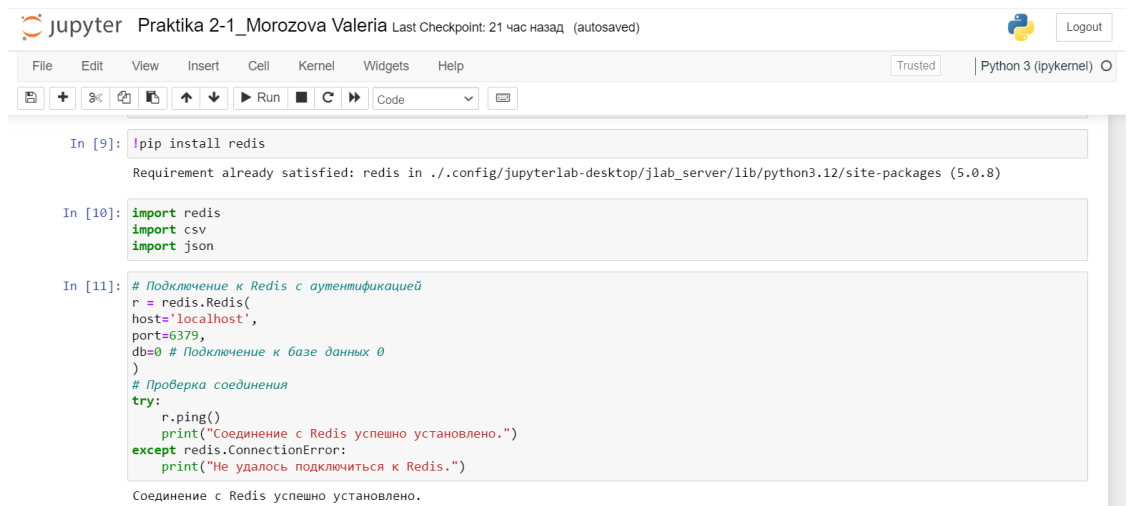
{'_id': ObjectId('66e3f618294ad1c0de522771'), 'lab_name': 'Lab 1', 'subject': 'Physics', 'date': '2024-08-28', 'score': 85}
{'_id': ObjectId('66e3f618294ad1c0de522772'), 'lab_name': 'Lab 2', 'subject': 'Chemistry', 'date': '2024-08-29', 'score': 90}
{'_id': ObjectId('66e3f618294ad1c0de522773'), 'lab_name': 'Lab 3', 'subject': 'Biology', 'date': '2024-08-30', 'score': 88}

In [8]: client.close()
```

6. Создание документа в коллекции и вывод данных



7. Успешное подключение к Redis



The screenshot shows a JupyterLab window titled "Praktika 2-1_Morozova Valeria". The interface includes a top bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help" menus. Below the menus is a toolbar with icons for file operations and a "Run" button. The main area contains three code cells. The first cell (In [9]) runs `!pip install redis`, resulting in the message "Requirement already satisfied: redis in ./config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (5.0.8)". The second cell (In [10]) imports `redis`, `csv`, and `json`. The third cell (In [11]) connects to a Redis instance on localhost:6379, db=0, and prints a confirmation message: "Соединение с Redis успешно установлено."

```
In [9]: !pip install redis

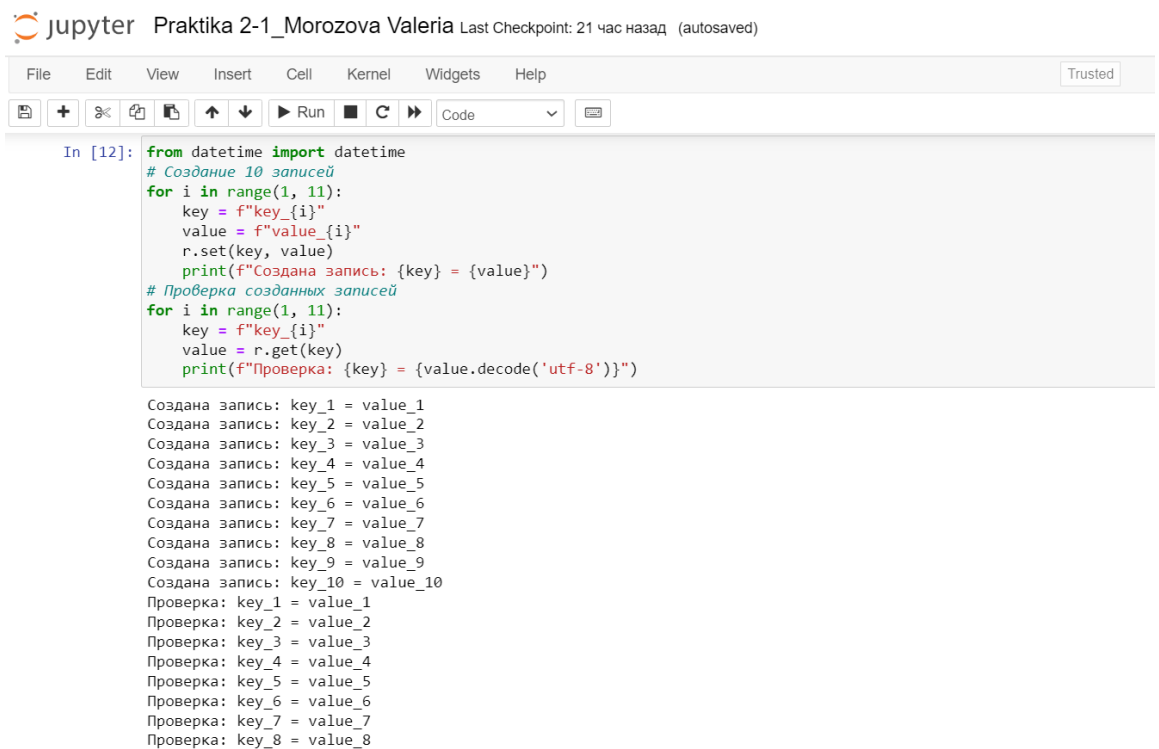
Requirement already satisfied: redis in ./config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (5.0.8)

In [10]: import redis
import csv
import json

In [11]: # 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.
```

8. Подключение к Redis в Jupyter

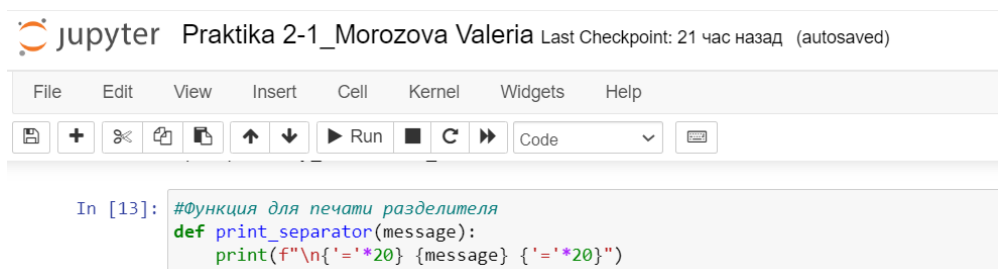


The screenshot shows a JupyterLab window titled "Praktika 2-1_Morozova Valeria". The interface includes a top bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help" menus. Below the menus is a toolbar with icons for file operations and a "Run" button. The main area contains one code cell (In [12]) that creates 10 records in Redis and then verifies them. The output shows the keys and values for each record, followed by the verification results.

```
In [12]: 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')}")

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
```

9. Создание 10 записей



The screenshot shows a JupyterLab window titled "Praktika 2-1_Morozova Valeria". The interface includes a top bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help" menus. Below the menus is a toolbar with icons for file operations and a "Run" button. The main area contains one code cell (In [13]) that defines a function `print_separator` for printing a separator line.

```
In [13]: #Функция для печати разделителя
def print_separator(message):
    print(f"\n{'='*20} {message} {'='*20}")
```

10. Создание функции для печати разделителя

```
File Edit View Insert Cell Kernel Widgets Help Trusted
In [14]: 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)}")

===== Создание данных =====
Создана строка: string_1 = value_1
Создана строка: string_2 = value_2
Создана строка: string_3 = value_3
```

11. Создание данных разного типа

```
File Edit View Insert Cell Kernel Widgets Help Trusted
In [15]: 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)}")

===== 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)]
```

12. Получение данных по ключу

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
In [16]: print_separator("Обновление данных")
r.set("string_1", "new_value")
print(f"Обновлена строка: string_1 = {r.get('string_1')}")
r.lset("list_1", 0, "new_item")
print(f"Обновлен список: list_1 = {r.lrange('list_1', 0, -1)}")
r.sadd("set_1", "new_element")
print(f"Обновлено множество: set_1 = {r.smembers('set_1')}")
r.hset("hash_1", "new_field", "new_value")
print(f"Обновлен хэш: hash_1 = {r.hgetall('hash_1')}")
r.zadd("zset_1", {"new_member": 5})
print(f"Обновлено упорядоченное множество: zset_1 = {r.zrange('zset_1', 0, -1, withscores=True)}")

===== 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'new_element', b'element_2'}
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)]
```

13. Обновление данных по ключу

```
In [17]: 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")
```

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

```
In [18]: 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)}")
```

```
===== 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
```

14. Удаление и проверка удаленных данных по ключу

```
In [19]: 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)
```

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

```
# Подключение к Redis
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]
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted
+ -> <-> ↺ ↻ ⬆ ⬇ ▶ Run ■ ↺ ▶ Code ▾
```

```

        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)])
# Закрываем соединения
r.close()
print(f"Данные сохранены в файл '{filename}'")

In [21]: # Выполнение загрузки
dump_redis_to_csv()

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

15. Выгрузка всех данных из Redis в csv

```

Данные сохранены в файл 'redis_dump.csv'

In [22]: ls

Desktop/      Pictures/      google-chrome-stable_current_amd64.deb
Documents/    Public/        mongoDB.ipynb
Downloads/    Templates/     mongodb/
MongoDB-Copy1.ipynb  Untitled.ipynb  pgredis/
MongoDB.ipynb  Untitled1.ipynb  redis_dump.csv
Music/         Videos/        snap/

In [ ]: Морозова Валерия
```

16. Проверка наличия нового вайла

Выполнение заданий по вариантам (№10)

```
Var#10_Morozova Valeria.ipynb
File Edit View Run Kernel Tabs Settings Help
+ -> <-> ↺ ↻ ⬆ ⬇ ▶ Run ■ ↺ ▶ Code ▾ Python 3 (ipykernel) ⌵
```

```

[ ]: #morozova

[1]: !pip install pymongo

Requirement already satisfied: pymongo in ./config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (4.8.0)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in ./config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from pymongo) (2.6.1)

[2]: from pymongo import MongoClient

[4]: mongo_uri = "mongodb://mongouser:mongopasswd@localhost:27017"

[5]: try:
# Подключение к MongoDB
client = MongoClient(mongo_uri)
# Проверка подключения
client.admin.command('ping')
print("Подключение к MongoDB установлено успешно!")
# Выбор базы данных
db = client['labs']
# Выбор коллекции
labs_collection = db['lab21']
except Exception as e:
print(f"Ошибка подключения: {e}")

Подключение к MongoDB установлено успешно!
```

17. Подключение к MongoDB в Jupyter


```
[27]: # Чтение CSV файла
csv_file = '/home/nosql/Downloads/Продукты и поставщики (1).csv'
data = pd.read_csv(csv_file, delimiter = ";")

[28]: data

[28]:
```

	Product Name	Category	Supplier	Origin Country	Last Delivery Date	Price
0	Gala Apple	Fruits	Fresh Farms Ltd.	USA	15.08.2023	0,52
1	Banana	Fruits	Tropicana Suppliers	Ecuador	16.08.2023	0,37
2	Carrot	Vegetables	Green Valley Organics	Netherlands	18.08.2023	0,41
3	Chicken Breast	Meat	Prime Meats Co.	USA	17.08.2023	0,91
4	Almond Milk	Beverages	NutriLife Foods	USA	21.08.2023	0,44
...
98	Ricotta Cheese	Dairy	Italian Cheese Co.	Italy	28.08.2023	0,05
99	Yellow Lentils	Grains	Legume World	India	29.08.2023	0,54
100	Lemon Juice	Beverages	Citrus Groves	Italy	30.08.2023	0,05
101	Gluten-Free Pasta	Grains	Italian Pasta Co.	Italy	30.08.2023	0,17
102	Organic Kale Chips	Snacks	Leafy Greens Ltd.	USA	29.08.2023	0,99

103 rows x 6 columns

```
[31]: try:
# Вставка данных в коллекцию
result = labs_collection.insert_many(data)
# Вывод идентификаторов вставленных документов
print("Данные успешно загружены в коллекцию 'labs'.")
print("Идентификаторы вставленных документов:", result.inserted_ids)
```

18. Чтение файла и загрузка данных в коллекцию

```
•[23]: # Создание новой коллекции from Morozova Valeria
collection_name = 'new_collection'
new_collection = db[collection_name]
```

19. Создание новой коллекции

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

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

[30]: print(df)

[{'product_name': 'Apple Gala', 'category': 'Fruit', 'supplier': 'Fresh Farms Ltd.', 'origin_country': 'USA', 'price_per_unit': 1.5, 'last_delivery_date': '2023-09-10', '_id': ObjectId('66f68afa0981396b99372dd5')}, {'product_name': 'Banana', 'category': 'Fruit', 'supplier': 'Tropicana Suppliers', 'origin_country': 'Ecuador', 'price_per_unit': 0.5, 'last_delivery_date': '2023-09-08', '_id': ObjectId('66f68afa0981396b99372dd6')}, {'product_name': 'Carrot', 'category': 'Vegetable', 'supplier': 'Green Valley', 'origin_country': 'Netherlands', 'price_per_unit': 0.8, 'last_delivery_date': '2023-09-12', '_id': ObjectId('66f68afa0981396b99372dd7')}, {'product_name': 'Chicken Breast', 'category': 'Meat', 'supplier': 'Prime Meats Co.', 'origin_country': 'USA', 'price_per_unit': 6.5, 'last_delivery_date': '2023-09-05', '_id': ObjectId('66f68afa0981396b99372dd8')}, {'product_name': 'Almond Milk', 'category': 'Beverage', 'supplier': 'NutriLife Foods', 'origin_country': 'USA', 'price_per_unit': 3.2, 'last_delivery_date': '2023-09-03', '_id': ObjectId('66f68afa0981396b99372dd9')}, {'product_name': 'Quinoa', 'category': 'Grains', 'supplier': 'Organic Grains Co.', 'origin_country': 'Peru', 'price_per_unit': 2.3, 'last_delivery_date': '2023-09-02', '_id': ObjectId('66f68afa0981396b99372dda')}, {'product_name': 'Cheddar Cheese', 'category': 'Dairy', 'supplier': 'Dairyland Ltd.', 'origin_country': 'UK', 'price_per_unit': 4.0, 'last_delivery_date': '2023-09-11', '_id': ObjectId('66f68afa0981396b99372ddb')}, {'product_name': 'Tomato', 'category': 'Vegetable', 'supplier': 'Farm Fresh Inc.', 'origin_country': 'Spain', 'price_per_unit': 1.1, 'last_delivery_date': '2023-09-06', '_id': ObjectId('66f68afa0981396b99372ddc')}, {'product_name': 'Basmati Rice', 'category': 'Grains', 'supplier': 'Asian Foods', 'origin_country': 'India', 'price_per_unit': 1.7, 'last_delivery_date': '2023-09-09', '_id': ObjectId('66f68afa0981396b99372ddd')}, {'product_name': 'Orange Juice', 'category': 'Beverage', 'supplier': 'Citrus Groves', 'origin_country': 'Brazil', 'price_per_unit': 2.5, 'last_delivery_date': '2023-09-07', '_id': ObjectId('66f68afa0981396b99372dde')}, {'product_name': 'Apple Gala', 'category': 'Fruit', 'supplier': 'Fresh Farms Ltd.', 'origin_country': 'USA', 'price_per_unit': 1.5, 'last_delivery_date': '2023-08-15', '_id': ObjectId('66f68afa0981396b99372ddf')}, {'product_name': 'Banana', 'category': 'Fruit', 'supplier': 'Tropicana Suppliers', 'origin_country': 'Ecuador', 'price_per_unit': 0.4, 'last_delivery_date': '2023-08-17', '_id': ObjectId('66f68afa0981396b99372dde0')}, {'product_name': 'Carrot', 'category': 'Vegetable', 'supplier': 'Green Valley Organics', 'origin_country': 'Netherlands', 'price_per_unit': 0.7, 'last_delivery_date': '2023-08-14', '_id': ObjectId('66f68afa0981396b99372dde1')}, {'product_name': 'Chicken Breast', 'category': 'Meat', 'supplier': 'Prime Meats Co.', 'origin_country': 'USA', 'price_per_unit': 5.8, 'last_delivery_date': '2023-08-20', '_id': ObjectId('66f68afa0981396b99372dde2')}, {'product_name': 'Almond Milk', 'category': 'Beverage', 'supplier': 'NutriLife Foods', 'origin_country': 'USA', 'price_per_unit': 2.9, 'last_delivery_date': '2023-08-18', '_id': ObjectId('66f68afa0981396b99372dde3')}, {'product_name': 'Whole Grain Quinoa', 'category': 'Grains', 'supplier': 'Organic Grains Co.', 'origin_country': 'Peru', 'price_per_unit': 2.6, 'last_delivery_date': '2023-08-22', '_id': ObjectId('66f68afa0981396b99372dde4')}]
```

20. Данные добавлены в коллекцию

```
[31]: documents = new_collection.find({'product_name': 'Apple Gala'})
      for doc in documents:
          print(doc)

{'_id': ObjectId('66f68afa0981396b99372dd5'), 'product_name': 'Apple Gala', 'category': 'Fruit', 'supplier': 'Fresh Farms Ltd.', 'origin_country': 'USA', 'price_per_unit': 1.5, 'last_delivery_date': '2023-09-10'}
{'_id': ObjectId('66f68afa0981396b99372ddf'), 'product_name': 'Apple Gala', 'category': 'Fruit', 'supplier': 'Fresh Farms Ltd.', 'origin_country': 'USA', 'price_per_unit': 1.5, 'last_delivery_date': '2023-08-15'}

[32]: documents = new_collection.find({'origin_country': 'Italy'})
      for doc in documents:
          print(doc)

{'_id': ObjectId('66f68afa0981396b99372deb'), 'product_name': 'Broccoli', 'category': 'Vegetable', 'supplier': 'Veggie Delight', 'origin_country': 'Italy', 'price_per_unit': 1.2, 'last_delivery_date': '2023-08-17'}
{'_id': ObjectId('66f68afa0981396b99372df0'), 'product_name': 'Pasta Fusilli', 'category': 'Grains', 'supplier': 'Italian Pasta Co.', 'origin_country': 'Italy', 'price_per_unit': 1.3, 'last_delivery_date': '2023-08-23'}
{'_id': ObjectId('66f68afa0981396b99372df2'), 'product_name': 'Mozzarella Cheese', 'category': 'Dairy', 'supplier': 'Cheese World', 'origin_country': 'Italy', 'price_per_unit': 4.3, 'last_delivery_date': '2023-08-27'}

[ ]:
```

21. Выполнение выборки по имени продукта и стране

Вывод:

1. При первом запросе видно, что Яблоки Гала завозились в разные дни, следовательно строки уникальны
2. Выяснилось, что из Италии завозятся несколько продуктов (Броccoli, Сыр Моцарелла и Паста Фетучини)

```
[35]: # Фильтр для нахождения документа
      filter_query = {'origin_country': 'Italy'}

      # Обновление данных
      update_query = {'$set': {'origin_country': 'Peru'}}

      # Выполнение обновления
      new_result = new_collection.update_one(filter_query, update_query)

      print(f'Обновлено документов: {new_result.modified_count}')
      Обновлено документов: 1

[36]: documents = new_collection.find({'product_name': 'Broccoli'})
      for doc in documents:
          print(doc)

{'_id': ObjectId('66f68afa0981396b99372deb'), 'product_name': 'Broccoli', 'category': 'Vegetable', 'supplier': 'Veggie Delight', 'origin_country': 'Peru', 'price_per_unit': 1.2, 'last_delivery_date': '2023-08-17'}

[ ]:
```

22. Обновление данных выполнено успешно

Вывод: как видно, Броccoli теперь поставляются из Перу

```
[38]: db.new_collection.delete_many({'product_name': 'Mozzarella Cheese'})

[38]: DeleteResult({'n': 1, 'ok': 1.0}, acknowledged=True)

[ ]: #Morozova
```

23. Удаление 1 документа (продукта сыр Моцарелла)

The screenshot shows the Visual Studio Code interface with the file `docker-compose.yml` open. The file contains the following configuration:

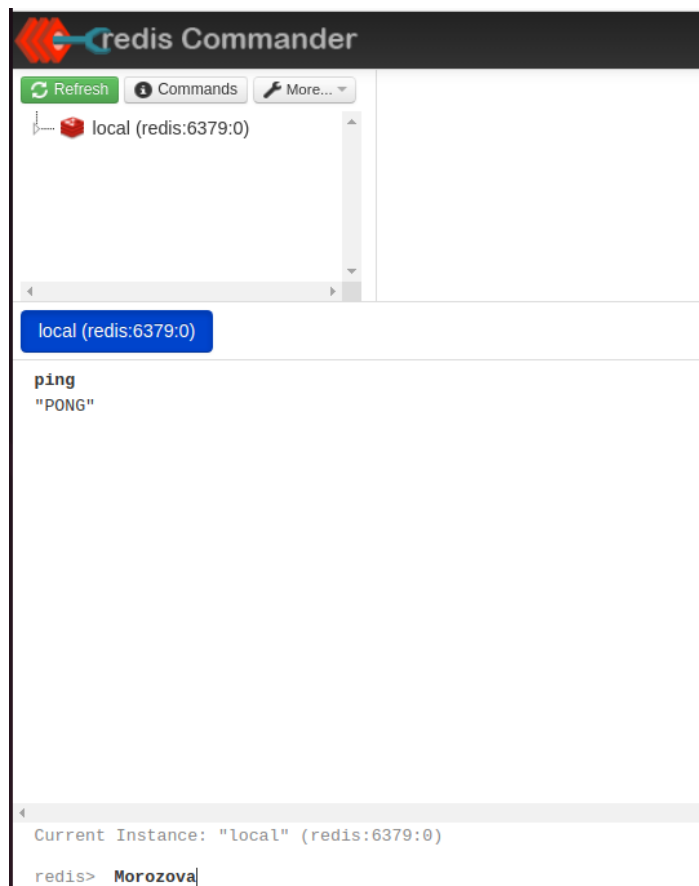
```
4 #redis://localhost:6379
5 #postgres://username:password@localhost:5435/database-name.
6 #Pgweb and Redis Commander are available at http://localhost:8085 and http://localhost:8086
7 services:
8   postgres:
9     image: postgres:alpine
10    environment:
11      POSTGRES_DB: database-name
12      POSTGRES_PASSWORD: password
13      POSTGRES_USER: username
14    ports:
15      - 5435:5432
```

The terminal window shows the execution of the command `sudo docker compose up -d` and the output:

```
nosql@nosql-vm:~/pgredis$ sudo docker compose up -d
[sudo] password for nosql:
[+] Running 4/4
✓ Container pgredis-redis-1      Started
✓ Container pgredis-redis-commander-1 Started
✓ Container pgredis-postgres-1   Started
✓ Container pgredis-pgweb-1      Started
nosql@nosql-vm:~/pgredis$
```

The terminal also shows the command `History restored` and the prompt `morozova`.

24. Открытие файла pgredis



25. Проверка подключения словом ping

```
Var10_Morozova V.ipynb
File Edit View Run Kernel Tabs Settings Help

[ ]: #Morozova

[39]: !pip install redis
Requirement already satisfied: redis in /home/nosql/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-package

[40]: 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.

[ ]:
```

26. Подключение к Redis в Jupyter

```
Var10_Morozova V.ipynb
File Edit View Run Kernel Tabs Settings Help

print("Не удалось подключиться к Redis.")

Соединение с Redis успешно установлено.

[42]: import pandas as pd
data = pd.read_csv('Продукты и поставщики (2).csv', delimiter=';')
print(data)

   ID_product  Product Name  Category  ID_supplier \
0           111    Gala Apple    Fruits         1199
1           112     Banana    Fruits         1200
2           113     Carrot  Vegetables         1201
3           114  Chicken Breast    Meat         1202
4           115   Almond Milk  Beverages         1203
..         ...           ...         ...         ...
98          209  Ricotta Cheese    Dairy         1297
99          210  Yellow Lentils  Grains         1298
100         211   Lemon Juice  Beverages         1299
101         212  Gluten-Free Pasta  Grains         1300
102         213  Organic Kale Chips  Snacks         1301

   Supplier Origin Country  Expiration date  Price
0    Fresh Farms Ltd.      USA    15.08.2023    0,52
1  Tropicana Suppliers    Ecuador    16.08.2023    0,37
2  Green Valley Organics  Netherlands    18.08.2023    0,41
3    Prime Meats Co.      USA    17.08.2023    0,91
4    NutriLife Foods      USA    21.08.2023    0,44
..         ...           ...         ...         ...
98  Italian Cheese Co.      Italy    28.08.2023    0,05
99    Legume World      India    29.08.2023    0,54
100   Citrus Groves      Italy    30.08.2023    0,05
101  Italian Pasta Co.      Italy    30.08.2023    0,17
102  Leafy Greens Ltd.      USA    29.08.2023    0,99

[103 rows x 8 columns]

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

27. Импорт данных из csv файла и создание функции разделителя

```
Var10_Morozova V.ipynb
File Edit View Run Kernel Tabs Settings Help

102      213 Organic Kale Chips      Snacks      1301

      Supplier Origin Country Expiration date Price
0      Fresh Farms Ltd.      USA      15.08.2023      0,52
1      Tropicana Suppliers      Ecuador      16.08.2023      0,37
2      Green Valley Organics      Netherlands      18.08.2023      0,41
3      Prime Meats Co.      USA      17.08.2023      0,91
4      Nutrilife Foods      USA      21.08.2023      0,44
..      ..      ..      ..      ..
98      Italian Cheese Co.      Italy      28.08.2023      0,05
99      Legume World      India      29.08.2023      0,54
100     Citrus Groves      Italy      30.08.2023      0,05
101     Italian Pasta Co.      Italy      30.08.2023      0,17
102     Leafy Greens Ltd.      USA      29.08.2023      0,99

[103 rows x 8 columns]

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

[60]: r.hmset(f'record:{index}', item_dict)

/tmp/ipykernel_6626/3935466372.py:1: DeprecationWarning: Redis.hmset() is deprecated. Use Redis.hset() instead.
r.hmset(f'record:{index}', item_dict)

[60]: True
```

28. Добавление данных в Redis

```
Var10_Morozova V.ipynb
File Edit View Run Kernel Tabs Settings Help

Данные успешно загружены в Redis.

[70]: #Используем уникальный ключ для каждой записи, например, id
a = r.get(27)
print(a)

b'Vene'
```

29. Выборка товара по ключу

```
Var10_Morozova V.ipynb
File Edit View Run Kernel Tabs Settings Help

b'Vene'

[71]: print_separator("Обновление данных")
r.set("27", "Jack Daniels Honey")

===== Обновление данных =====

[71]: True

[66]: a = r.get(27)
print(a)

b'Jack Daniels Honey'
```

30. Заменяем значение и обновляем данные

```
Var10_Morozova V.ipynb
File Edit View Run Kernel Tabs Settings Help

[72]: r.delete(27)

[72]: 1

[73]: print_separator("Проверка удаленных данных")
      a = r.get(27)
      print(a)

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

31. Удаление записи и проверка

Вывод: данные из файла были успешно загружены, обновлены и по выборке один из товаров был удален

Выполнение Cypher-запроса на графовой базе Neo4j

console.neo4j.org

```
(0:Crew {name:"Neo"}) [(0)-[0:KNOWS]->(1), (1)-[2:KNOWS]->(2)] (2:Crew {name:"Trinity"})
(0:Crew {name:"Neo"}) [(0)-[0:KNOWS]->(1), (1)-[3:KNOWS]->(3)] (3:Crew:Matrix {name:"Cypher"})
(0:Crew {name:"Neo"}) [(0)-[0:KNOWS]->(1), (1)-[3:KNOWS]->(3), (3)-[4:KNOWS]->(4)] (4:Matrix {name:"Agent Smith"})
```

Query took 20 ms and returned 4 rows. [Result Details](#)

You can modify and query this graph by entering statements in the input field at the bottom. For some syntax help hit the [Help](#) button. If you want to share your graph, just do it with [Share](#)

Query:

```
CREATE (C:course{name:'Discrete Mathematics'}), (S:course{name:'Databases'}), (I:course{name:'Data Processing'}), (N:person{name:'Natalia'}), (V:person{name:'Victoria'}), (O:person{name:'Olga'}), (St:person{name:'Katarina'}), (Dm:person{name:'Dmitry'}), (K:person{name:'Katarina'}), (Elena:student{name:'Teresa'}), (Nina:student{name:'Nina'}), (Victor:student{name:'Victor'}), (Olga:student{name:'Olga'}), (Stas:student{name:'Stas'}), (Anna:student{name:'Anna'}), (Konstantin:student{name:'Konstantin'}), (Ilia:student{name:'Ilia'}), (E:author)->(C), (N)->(C), (A)->(C), (A)->(S), (A)->(I), (I)->(S), (I)->(I), (K)->(S), (Elena)->(I), (Elena)->(C), (Nina)->(S), (Nina)->(I), (Victor)->(S), (Victor)->(I), (Olga)->(S), (Olga)->(I), (Stas)->(S), (Stas)->(I), (Stas)->(S), (Stas)->(I), (Anna)->(S), (Anna)->(I), (Alexandr)->(S), (Alexandr)->(I), (Dmitry)->(S), (Dmitry)->(I), (Konstantin)->(S), (Konstantin)->(I), (Roman)->(S), (Roman)->(I), (Ilia)->(S), (Ilia)->(I)
```

Query took 35 ms and returned no rows.
Updated the graph - created 25 nodes and 37 relationships set 25 properties [Result Details](#)

Граф, построенный на основе исходного запроса

10. Напишите запрос, который вернет количество студентов, записанных на каждый из курсов

- ```
Query:
CREATE (:C:course{name:'Discrete Mathematics'}), (:S:course{name:'Databases'}), (:I:course{name:'Data Processing'}), (:E:person{name:'Elena'}),
(:N:person{name:'Natalia'}), (:V:person{name:'Victoria'}), (:O:person{name:'Olga'}), (:St:person{name:'Stas'}), (:A:person{name:'Alex'}), (:D:person{name:'Dan'}),
(:Al:person{name:'Alex'}), (:Dm:person{name:'Dmitry'}), (:K:person{name:'Katarina'}), (Elena:student{name:'Ter'}, (Nina:student{name:'Nina'}),
(Victoria:student{name:'Victor'}, (Olga:student{name:'Olga'}), (Stas:student{name:'Stas'}), (Anna:student{name:'Anna'}), (Dmitry:student{name:'Denis'}),
(Alexander:student{name:'Alexander'}, (Dmitry:student{name:'Dmitry'}), (Konstantin:student{name:'Konstantin'}), (Roman:student{name:'Roman'}),
(Ilia:student{name:'Ilia'}), (E)-[:author]->(C), (N)-[:author]->(C), (A)-[:author]->(S), (A)-[:author]->(I), (AL)-[:author]->(I), (DM)-[:author]->(I), (O)-
[:speaker]->(C), (St)-[:editor]->(C), (V)-[:designer]->(C), (D)-[:speaker]->(S), (St)-[:editor]->(S), (I)-[:designer]->(S), (DM)-[:speaker]->(I), (St)-[:editor]->
(I), (K)-[:designer]->(I), (Elena)-[:learn]->(I), (Elena)-[:learn]->(C), (Nina)-[:learn]->(S), (Nina)-[:learn]->(C), (Victoria)-[:learn]->(C), (Victoria)-[:learn]->
(C), (Olga)-[:learn]->(S), (Olga)-[:learn]->(C), (Stas)-[:learn]->(I), (Stas)-[:learn]->(C), (Stas)-[:learn]->(S), (Stas)-[:learn]->(C), (Stas)-[:learn]->(C),
(Anna)-[:learn]->(S), (Denis)-[:learn]->(I), (Alexander)-[:learn]->(C), (Dmitry)-[:learn]->(I), (Dmitry)-[:learn]->(C), (Konstantin)-[:learn]->(S), (Roman)-[:learn]->(C), (Ilia)-[:learn]->(C)
```

Query took 34 ms and returned no rows.

Updated the graph - created 25 nodes and 37 relationships set 25 properties

```
Result Details
```

```
Query:
MATCH S:(student)-[:ENROLLED_IN]->(C:course) RETURN c.name AS CourseName, COUNT(s) AS NumberOfStudents
Error: org.neo4j.graphdb.QueryExecutionException: Invalid input ':': expected an identifier character, whitespace, NodeLabel, a property map, a relationship pattern, ':', USING, WHERE, FROM GRAPH, CONSTRUCT, LOAD CSV, START, MATCH, UNWIND, MERGE, CREATE UNIQUE, CREATE, SET, DELETE, REMOVE, FOREACH, WITH, CALL, RETURN, UNION, ':', or end of input (Line 1, column 24 (offset: 23))
"MATCH S:(student)-[:ENROLLED_IN]->(C:course) RETURN c.name AS CourseName, COUNT(s) AS NumberOfStudents ORDER BY Cou"
^
```

```
Query:
MATCH S:(student)-[:ENROLLED_IN]->(C:course) RETURN c.name AS CourseName, COUNT(s) AS NumberOfStudents ORDER BY CourseName
Error: org.neo4j.graphdb.QueryExecutionException: Invalid input ':': expected an identifier character, whitespace, NodeLabel, a property map, a relationship pattern, ':', USING, WHERE, FROM GRAPH, CONSTRUCT, LOAD CSV, START, MATCH, UNWIND, MERGE, CREATE UNIQUE, CREATE, SET, DELETE, REMOVE, FOREACH, WITH, CALL, RETURN, UNION, ':', or end of input (Line 1, column 24 (offset: 23))
"MATCH S:(student)-[:ENROLLED_IN]->(C:course) RETURN c.name AS CourseName, COUNT(s) AS NumberOfStudents ORDER BY CourseName"
^
```

```
MATCH S:(student)-[:ENROLLED_IN]->(C:course) RETURN c.name AS CourseName, COUNT(s) AS NumberOfStudents ORDER BY CourseName
MOROZOVA
```

- ❖ Имеет специальный Cypher язык запросов для создания графовых баз и манипуляций над ними
- ❖ Запрос на выборку данных из графа – это обход графа, осуществляется быстрее чем в реляционных моделях