

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 №4-1**

**Тема:**

«Сравнение подходов хранения больших данных»

Выполнил(а): Морозова Валерия АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2024

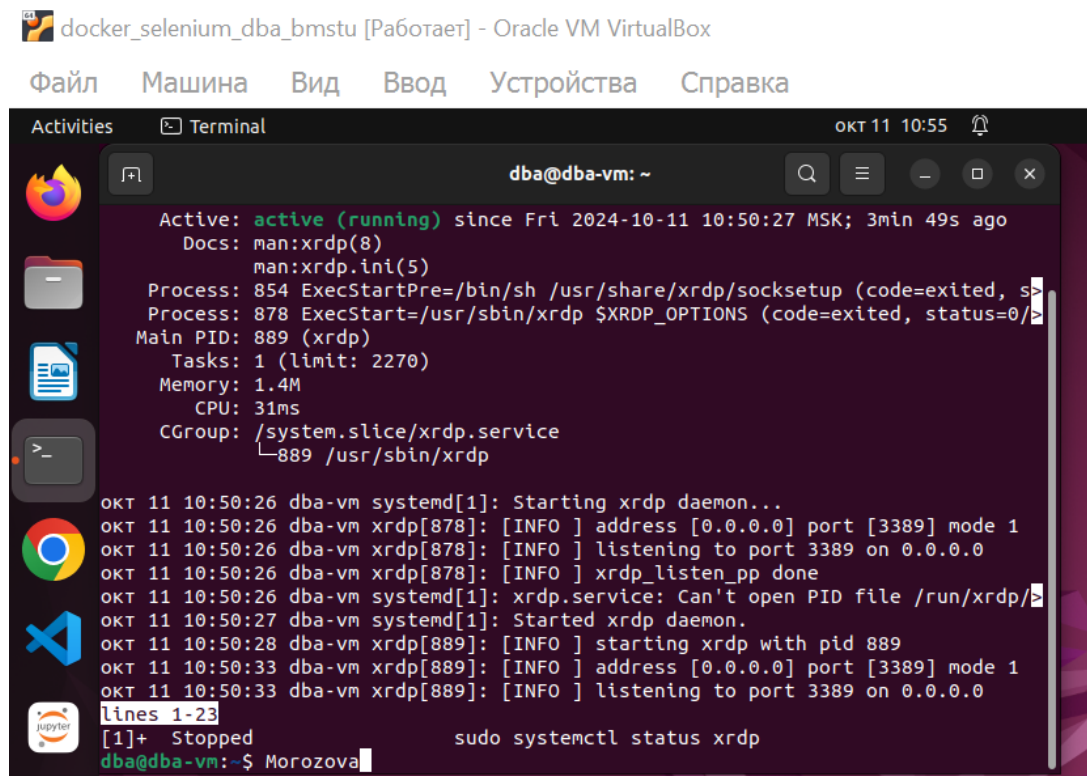
**Цель работы:** сравнить производительность и эффективность различных подходов к хранению и обработке больших данных на примере реляционной базы данных PostgreSQL и документо ориентированной базы данных MongoDB.

### Оборудование и программное обеспечение

- Компьютер с операционной системой Ubuntu.
- PostgreSQL.
- MongoDB.
- Python 3.x.
- Библиотеки: psycorp2, pymongo, pandas, matplotlib

### Практика на паре

Для начала в системе Ubuntu с помощью функции `sudo systemctl status xrdp` в терминале проверили статус службы удаленного рабочего стола.



```
docker_selenium_dba_bmstu [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Activities  Terminal  окт 11 10:55
dba@dba-vm: ~
Active: active (running) since Fri 2024-10-11 10:50:27 MSK; 3min 49s ago
Docs: man:xrdp(8)
      man:xrdp.ini(5)
Process: 854 ExecStartPre=/bin/sh /usr/share/xrdp/socksetup (code=exited, status=0)
Process: 878 ExecStart=/usr/sbin/xrdp $XRDPOPTIONS (code=exited, status=0)
Main PID: 889 (xrdp)
Tasks: 1 (limit: 2270)
Memory: 1.4M
CPU: 31ms
CGroup: /system.slice/xrdp.service
        └─889 /usr/sbin/xrdp

окт 11 10:50:26 dba-vm systemd[1]: Starting xrdp daemon...
окт 11 10:50:26 dba-vm xrdp[878]: [INFO ] address [0.0.0.0] port [3389] mode 1
окт 11 10:50:26 dba-vm xrdp[878]: [INFO ] listening to port 3389 on 0.0.0.0
окт 11 10:50:26 dba-vm xrdp[878]: [INFO ] xrdp_listen_pp done
окт 11 10:50:26 dba-vm systemd[1]: xrdp.service: Can't open PID file /run/xrdp/
окт 11 10:50:27 dba-vm systemd[1]: Started xrdp daemon.
окт 11 10:50:28 dba-vm xrdp[889]: [INFO ] starting xrdp with pid 889
окт 11 10:50:33 dba-vm xrdp[889]: [INFO ] address [0.0.0.0] port [3389] mode 1
окт 11 10:50:33 dba-vm xrdp[889]: [INFO ] listening to port 3389 on 0.0.0.0

lines 1-23
[1]+  Stopped
dba@dba-vm:~$ sudo systemctl status xrdp
dba@dba-vm:~$ Morozova
```

Рисунок 1. Выполнение запроса в терминале на VM

Затем нужно подключиться через удаленный рабочий стол к виртуальной машине. Проверяем айпи адрес, он должен соответствовать тому, который на телефоне и с которого осуществляется раздача интернета.

## 🏠 Redmi Note 9 Pro

### Свойства

SSID:	Redmi Note 9 Pro
Протокол:	Wi-Fi 4 (802.11n)
Тип безопасности:	WPA2-Personal
Диапазон сети:	2,4 ГГц
Канал сети:	6
Скорость линии (прием и передача):	144/144 (Mbps)
IPv6-адрес:	2a00:1fa0:4331:184c:3911:5e45:ed52:1a06
Локальный IPv6-адрес канала:	fe80::46ee:ccc4:6503:9dc4%18
IPv4-адрес:	192.168.219.236
DNS-серверы IPv4:	192.168.219.82
Изготовитель:	Intel Corporation
Описание:	Intel(R) Wi-Fi 6 AX201 160MHz
Версия драйвера:	22.50.0.7
Физический адрес (MAC):	A4-6B-B6-9A-50-3D

Копировать

Рисунок 2. Определение IPv4-адреса телефона

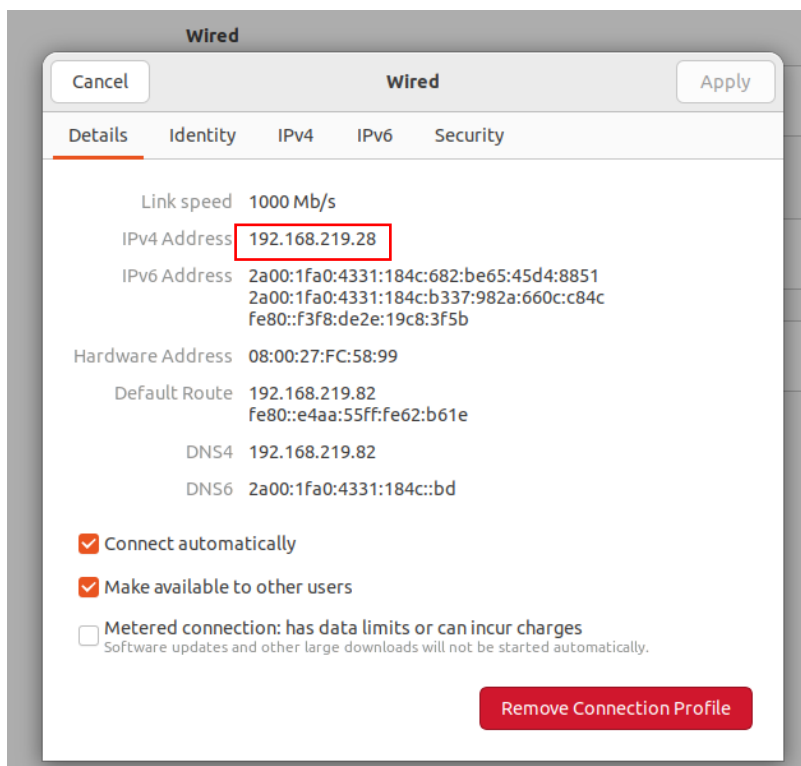


Рисунок 3. Проверка IPv4-адреса виртуальной машины

После того, как подключение к удаленному рабочему столу виртуальной машины успешно установлено, надо запустить контейнеры на основе настроек, определенных в файле docker-compose.yml (рисунок 4).

```
dba@dba-vm:~/Downloads/bachelor/course_4$ sudo docker compose up -d
[sudo] password for dba:
[+] Running 7/7
 ✓ Network course_4_backend          Created
 ✓ Volume "course_4_mongo_express_volume" Created
 ✓ Volume "course_4_mongodb_volume"  Created
 ✓ Container mongodb                  Started
 ✓ Container postgres_container       Started
 ✓ Container pgadmin_container        Started
 ✓ Container mongo-express            Started
```

Рисунок 4. Запуск контейнеров

Далее с помощью команды `sudo docker inspect (id_container)` надо узнать IP address сервера PgAdmin (рисунок 5).

```
dba@dba-vm:~/Downloads/bachelor/course_4$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
67db0315d272   dpape/pgadmin4 "/entrypoint.sh"        14 minutes ago Up 14 minutes 443/tcp, 0.0.0.0:5050->80/tcp, [::]:5050->80/t
6c1cd2565458   mongo-express:latest "/sbin/tini -- /dock..." 14 minutes ago Up 14 minutes 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
431f22e8b45d   postgres      "docker-entrypoint.s..." 14 minutes ago Up 14 minutes 0.0.0.0:5432->5432/tcp, :::5432->5432/tcp
964e2319d75e   mongo:latest  "docker-entrypoint.s..." 14 minutes ago Up 14 minutes 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp

dba@dba-vm:~/Downloads/bachelor/course_4$ sudo docker inspect 431f22e8b45d
[
  {
    "Id": "431f22e8b45df54d01497f0dec26ee00e2c89841557596680b4bc2ec50e370fb",
    "Created": "2024-10-11T08:14:07.007271392Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
      "postgres"
    ],
    "State": {
      "EndpointID": "",
      "Gateway": "",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "IPAddress": "",
      "IPPrefixLen": 0,
      "IPv6Gateway": "",
      "MacAddress": "",
      "Networks": {
        "course_4_backend": {
          "IPAMConfig": null,
          "Links": null,
          "Aliases": [
            "postgres_container",
            "postgres-compose"
          ],
          "MacAddress": "02:42:ac:13:00:02",
          "DriverOpts": null,
          "NetworkID": "cfdad8fc49c3594651389f9edfc810196a4c92563cd4b823b81150395c0dd6ad",
          "EndpointID": "deed568b309d13748d2cf226d0a36207e54cb5f697bb0b6e8e3e08cdca853b04",
          "Gateway": "172.19.0.1",
          "IPAddress": "172.19.0.2",
          "IPPrefixLen": 16,
          "IPv6Gateway": "",
          "GlobalIPv6Address": "",
          "GlobalIPv6PrefixLen": 0,
          "DNSNames": [
            "postgres_container",
            "postgres-compose",
            "431f22e8b45d"
          ]
        }
      }
    }
  }
]
```

Рисунок 5. "IPAddress" сервера PgAdmin : "172.19.0.2"

Теперь необходимо проверить подключение к Mongo Express.

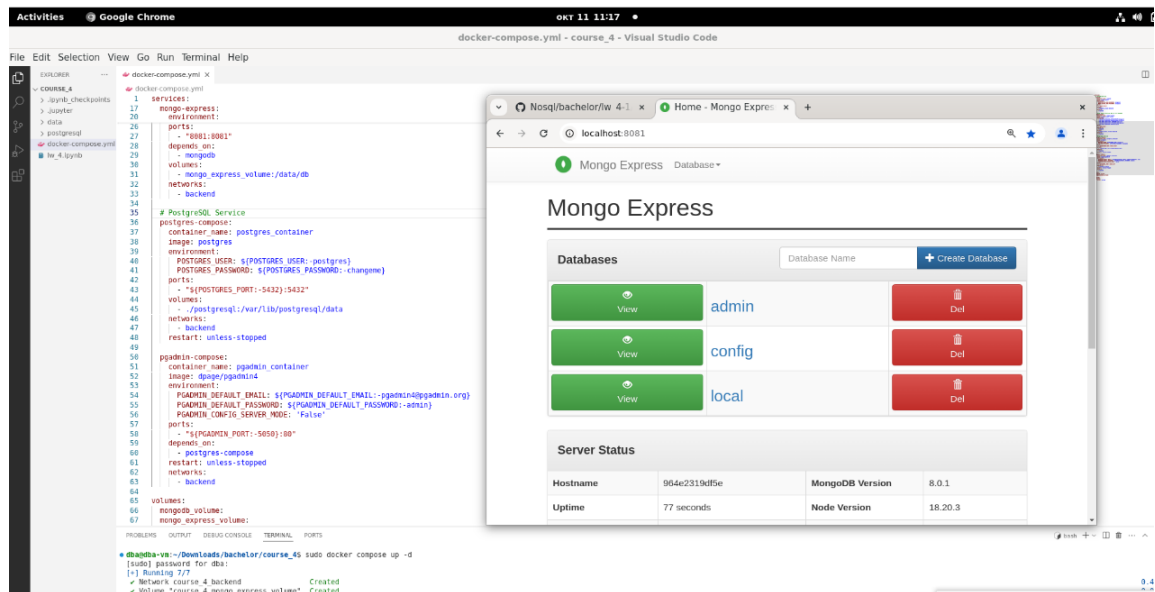


Рисунок 6. Подключение к Mongo Express установлено

Так же проверяем подключение к PgAdmin.

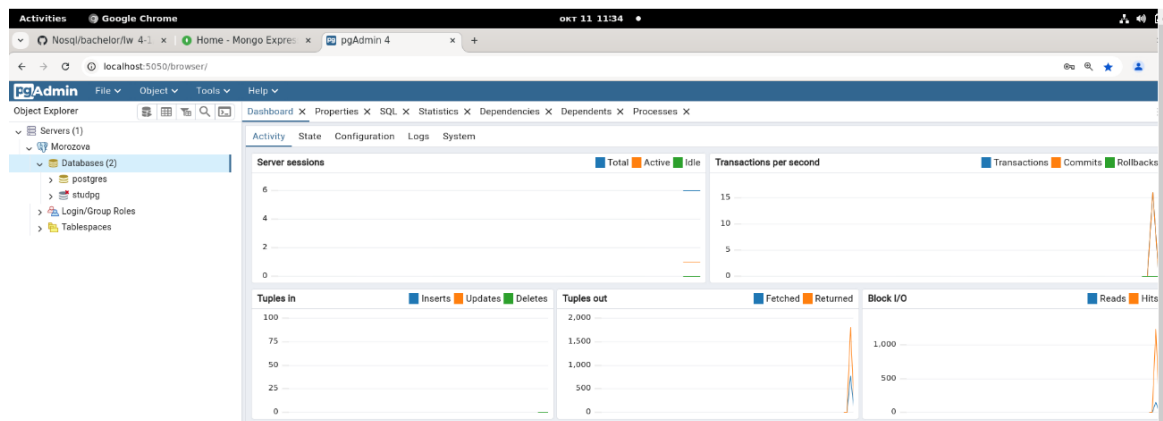


Рисунок 7. Успешное подключение к PgAdmin 4

Далее необходимо выполнить несколько запросов в Jupyter Notebook, в результате которых была создана таблица (рисунок 8) и построилась гистограмма (рисунок 9).

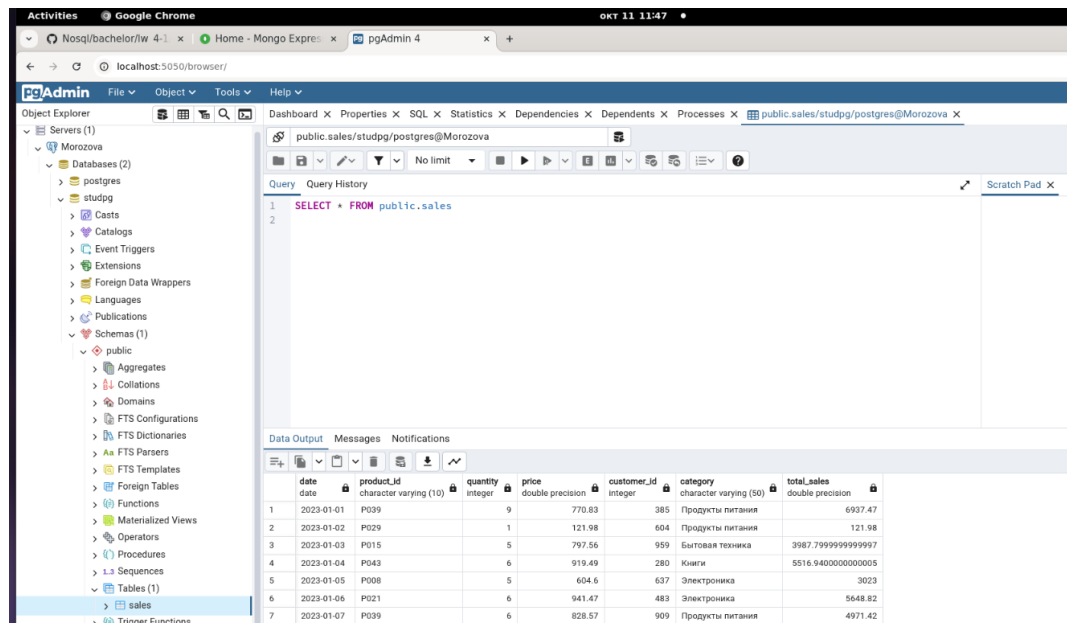


Рисунок 8. Отображение содержания новой таблицы sales в PgAdmin 4

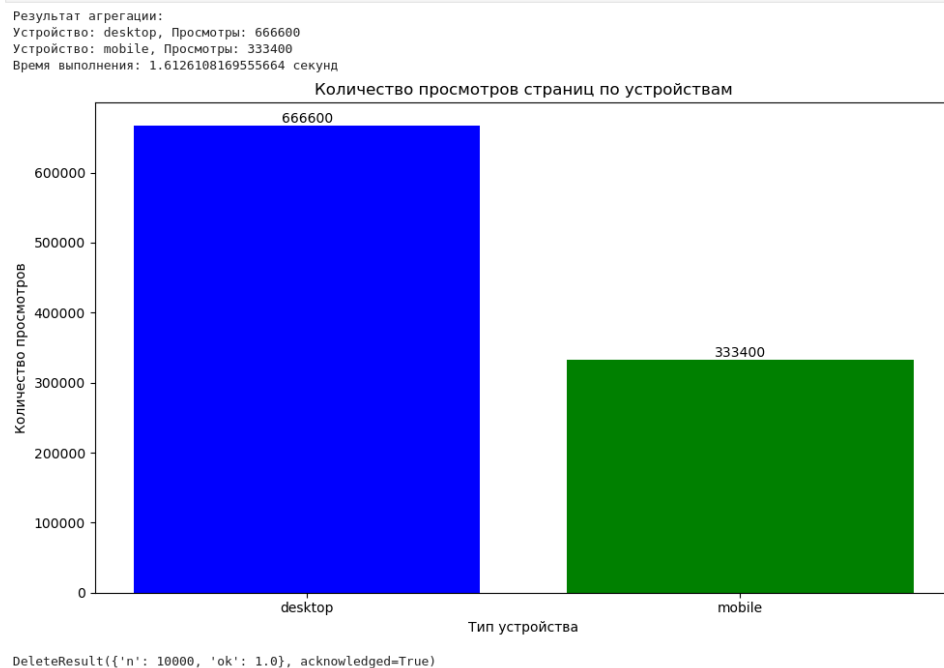


Рисунок 9. Результат выполнения запросов в Jupyter Notebook

## Индивидуальное задание

Вариант 10. Исследовать производительность при выполнении операций соединения (join) в реляционной и документо-ориентированной базах данных на примере системы управления персоналом.

В первую очередь устанавливаем модуль Faker (рисунок 10), с помощью которого далее будут сгенерированы фейковые данные.

```
[2]: !pip install faker

Collecting faker
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ConnectTimeoutError(<pip._vendor.urllib3.connection.HTTPSConnection object at 0x7045f3f79c70>, 'Connection to files.pythonhosted.org timed out. (connect timeout=15)')': /packages/64/82/f7d0c0a4ab512fd1572a315eec903d50a578c75d5aa894cf3f5cc04025e5/Faker-30.8.2-py3-none-any.whl.metadata
  Downloading Faker-30.8.2-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: python-dateutil>=2.4 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from faker) (2.9.0)
Requirement already satisfied: typing-extensions in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from faker) (4.12.2)
Requirement already satisfied: six>=1.5 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from python-dateutil>=2.4->faker) (1.16.0)
Downloading Faker-30.8.2-py3-none-any.whl (1.8 MB)
----- 1.8/1.8 MB 2.4 MB/s eta 0:00:00a 0:00:01

Installing collected packages: faker
Successfully installed faker-30.8.2

[ ]: #MOROZOVA VALERIA
```

Рисунок 10. Установка модуля faker

```
[1]: !pip install pandas numpy pymongo pycopg2-binary sqlalchemy

Requirement already satisfied: pandas in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (2.2.2)
Requirement already satisfied: numpy in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (2.1.0)
Requirement already satisfied: pymongo in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (4.10.1)
Requirement already satisfied: pycopg2-binary in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (2.9.9)
Requirement already satisfied: sqlalchemy in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (2.0.35)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from pandas) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from pymongo) (2.7.0)
Requirement already satisfied: typing-extensions>=4.6.0 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from sqlalchemy) (4.12.2)
Requirement already satisfied: greenlet!=0.4.17 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from sqlalchemy) (3.1.1)
Requirement already satisfied: six>=1.5 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

[ ]: #MOROZOVA VALERIA
```

Рисунок 11. Установка других модулей

Далее необходимо импортировать все необходимые библиотеки и проверить подключение к MongoDB и PostgreSQL.

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

[3]: import pandas as pd
import numpy as np
from pymongo import MongoClient
import psycopg2
from sqlalchemy import create_engine
from datetime import datetime, timedelta

[4]: def check_mongo_connection(client):
    try:
        # Проверка подключения к серверу
        client.server_info()
        print("Успешное подключение к MongoDB")
        return True
    except Exception as e:
        print(f"Ошибка подключения к MongoDB: {e}")
        return False

def check_postgres_connection(conn_params):
    try:
        conn = psycopg2.connect(**conn_params)
        print("Успешное подключение к PostgreSQL")
        return conn
    except Exception as e:
        print(f"Ошибка подключения к PostgreSQL: {e}")
        return None
```

Рисунок 12. Импорт библиотек и проверка подключения к серверу

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

[5]: from faker import Faker
import random

n_records = 1000

def generate_hr_data(n_records):
    employees = []
    fake = Faker()
    for _ in range(n_records):
        employee = {
            "employee_id": fake.uuid4(),
            "name": fake.name(),
            "department": fake.job(),
            "position": fake.job(),
            "hire_date": fake.date_this_decade(),
            "salary": random.randint(30000, 150000),
            "performance_score": round(random.uniform(1, 5), 1)
        }
        employees.append(employee)
    return employees

df_employees = pd.DataFrame(generate_hr_data(1000))
print(df_employees)
```

	employee_id	name \
0	c0f094ba-a652-48fb-a456-304d53667c14	Hayley Carey
1	4351d290-9635-4c4b-ad52-4e01d60509f8	Chad Shaffer
2	ea8cc4ca-1c23-48d3-9360-7d037d8eb2b1	Tracey Mitchell
3	35bbd5b1-361e-48d7-ac54-800de46ee92d	Richard Wade
4	f0aecd97-1f70-45e4-99d2-1960da1ee1a0	Rose Hoffman
...	...	...
995	1f6714c4-e473-4345-9de5-0a4a6ad70947	Ashley Russell
996	db0b17cb-2e3a-4103-89c1-a862ebb20d43	Jeremy McLaughlin
997	babcf61a-5cb3-4576-8e4c-a045996e184d	Tracy Brown
998	3bd5231f-e6cd-4290-b718-471e463178d9	James Preston
999	6a208781-a74b-4af1-846f-f84af0229ae1	Christopher Smith

Рисунок 13. Генерация данных по сотрудникам



```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

[11]: n_records = 1000

def generate_department_data(n_records):
    departments = []
    fake = Faker()
    for _ in range(n_records):
        department = {
            "department_id": fake.uuid4(),
            "department_name": fake.job(),
        }
        departments.append(department)
    return departments

df_departments = pd.DataFrame(generate_hr_data(1000))
print(df_departments)

      department_id      department_name
0  970375ce-a2b9-4097-98c6-4a8d6234fb3e  Advertising art director
1  8411f58c-369d-4508-a917-673267727687      Neurosurgeon
2  78c85576-d8ba-4437-9015-e924aac8a238  Financial controller
3  931ff0c2-0afc-4229-8326-3cdf9ed0e8bc  Broadcast presenter
4  3bdd09d6-249a-4461-a4a3-4e00f73fea73      Administrator
..      ...
995 b690ce93-a3de-4cc4-b6ec-9797ef5f8e90  Public house manager
996 b57ac627-9141-4526-9262-7bd6539e5659      Tax adviser
997 c807101c-8d6e-463f-9f86-0e3de8b7a5cd  Community arts worker
998 9f26c06e-7963-490f-8a74-fd5b6e069778      Fashion designer
999 827e456b-3096-435b-8b5a-63b21ea37285  Publishing rights manager

[1000 rows x 2 columns]

[ ]: #VALERIA MOROZOVA
```

Рисунок 14. Генерация данных по департаментам

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

Python 3 (ipykernel)

[6]: df = pd.DataFrame(df_employees)
df

      employee_id      name      department      position      hire_date      salary      performance_score
0  c0f094ba-a652-48fb-a456-304d53667c14  Hayley Carey      Social worker      Printmaker      2023-09-07      78237      1.7
1  4351d290-9635-4c4b-ad52-4e01d60509f8  Chad Shaffer      Buyer, industrial      Learning disability nurse      2022-03-12      42090      1.8
2  ea8cc4ca-1c23-48d3-9360-7d037d8eb2b1  Tracey Mitchell      Water engineer      Geneticist, molecular      2021-07-19      110185      3.9
3  35bbd5b1-361e-48d7-ac54-800de46ee92d  Richard Wade      Commercial art gallery manager      Solicitor, Scotland      2021-05-11      51341      2.3
4  f0aec9d97-1f70-45e4-99d2-1960da1ee1a0  Rose Hoffman      Engineering geologist      Medical illustrator      2020-12-20      113610      3.6
...      ...      ...      ...      ...      ...      ...
995 1f6714c4-e473-4345-9de5-0a4a6ad70947  Ashley Russell      Engineer, manufacturing      Teacher, secondary school      2020-10-19      79218      1.6
996 db0b17cb-2e3a-4103-89c1-a862ebb20d43  Jeremy McLaughlin      Tour manager      Engineer, energy      2023-07-23      61574      3.2
997 babc761a-5cb3-4576-8e4c-a045996e184d  Tracy Brown      Animal technologist      Automotive engineer      2021-04-22      139957      3.4
998 3bd5231f-e6cd-4290-b718-471e463178d9  James Preston      Museum/gallery conservator      Psychologist, counselling      2020-02-02      84974      2.9
999 6a208781-a74b-4af1-846f-f84af0229ae1  Christopher Smith      Music tutor      Surveyor, mining      2021-11-30      149975      3.8

1000 rows x 7 columns

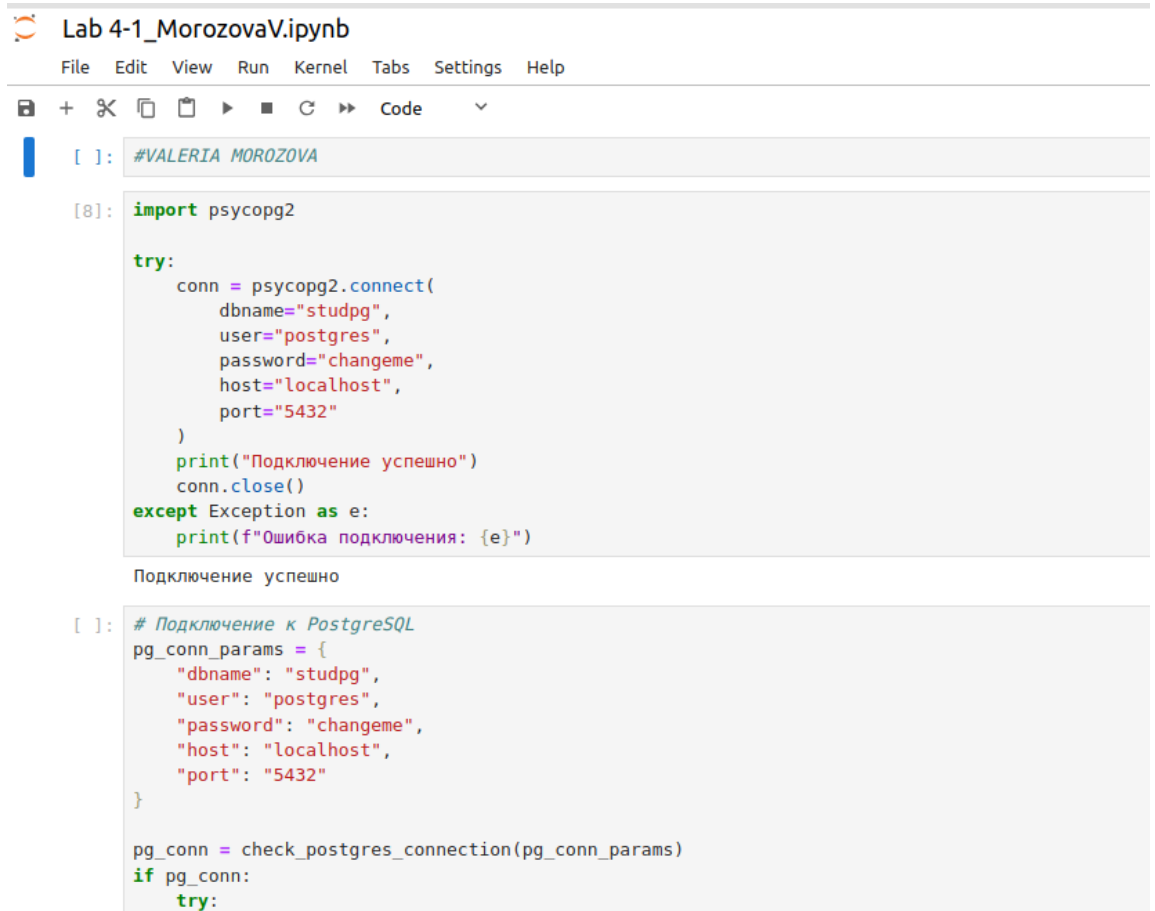
[7]: # Сохранение в CSV для дальнейшего использования
df.to_csv('data_employees.csv', index=False)

print("Данные сгенерированы и сохранены в data_employees.csv")
Данные сгенерированы и сохранены в data_employees.csv

[ ]: #MOROZOVA VALERIA
```

Рисунок 15. Преобразование данных в DataFrame и сохранение в файл csv

# PostgreSQL



The screenshot shows a Jupyter Notebook titled "Lab 4-1\_MorozovaV.ipynb". The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help) and a toolbar with icons for file operations and execution. The code is written in a light blue editor. The first cell contains a comment "#VALERIA MOROZOVA". The second cell, labeled "[8]:", contains Python code using the psycopg2 library to connect to a PostgreSQL database. The code defines connection parameters (dbname="studpg", user="postgres", password="changeme", host="localhost", port="5432") and attempts to connect. A successful connection is confirmed by the output "Подключение успешно". The third cell, labeled "[ ]:", contains code to define connection parameters in a dictionary and call a function "check\_postgres\_connection".

```
[ ]: #VALERIA MOROZOVA

[8]: import psycopg2

try:
    conn = psycopg2.connect(
        dbname="studpg",
        user="postgres",
        password="changeme",
        host="localhost",
        port="5432"
    )
    print("Подключение успешно")
    conn.close()
except Exception as e:
    print(f"Ошибка подключения: {e}")

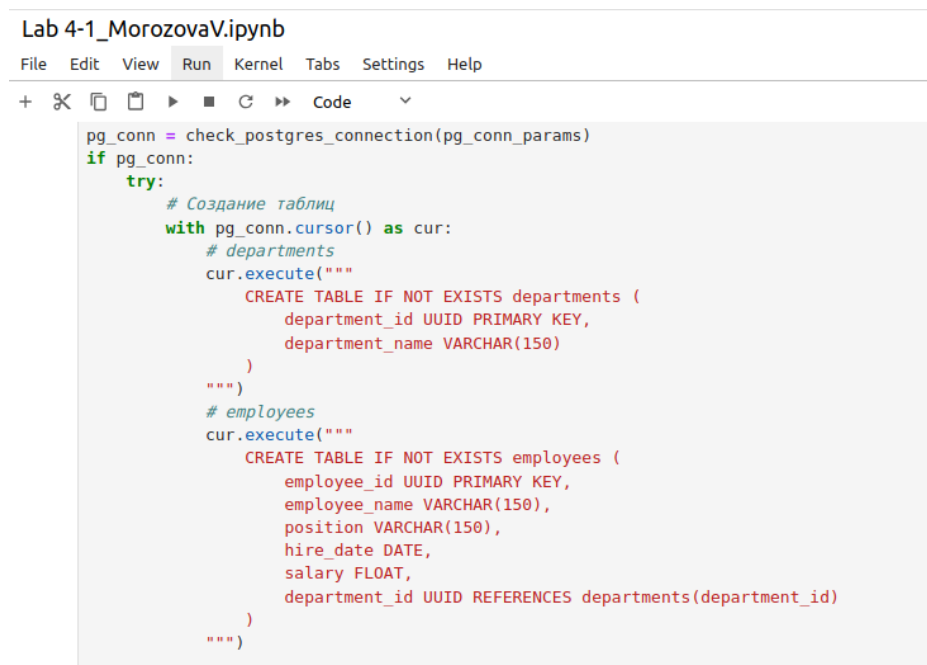
Подключение успешно

[ ]: # Подключение к PostgreSQL
pg_conn_params = {
    "dbname": "studpg",
    "user": "postgres",
    "password": "changeme",
    "host": "localhost",
    "port": "5432"
}

pg_conn = check_postgres_connection(pg_conn_params)
if pg_conn:
    try:
```

Рисунок 16. Успешное подключение к PostgreSQL

После подключения нужно создать две таблицы, в которые в последующем будут загружены ранее сгенерированные данные.



The screenshot shows the same Jupyter Notebook, but the "Run" button in the menu bar is highlighted. The code in the cell continues from the previous one, showing the creation of two tables: "departments" and "employees". The "departments" table has columns "department\_id" (UUID PRIMARY KEY) and "department\_name" (VARCHAR(150)). The "employees" table has columns "employee\_id" (UUID PRIMARY KEY), "employee\_name" (VARCHAR(150)), "position" (VARCHAR(150)), "hire\_date" (DATE), "salary" (FLOAT), and "department\_id" (UUID REFERENCES departments(department\_id)).

```
pg_conn = check_postgres_connection(pg_conn_params)
if pg_conn:
    try:
        # Создание таблиц
        with pg_conn.cursor() as cur:
            # departments
            cur.execute("""
                CREATE TABLE IF NOT EXISTS departments (
                    department_id UUID PRIMARY KEY,
                    department_name VARCHAR(150)
                )
            """)
            # employees
            cur.execute("""
                CREATE TABLE IF NOT EXISTS employees (
                    employee_id UUID PRIMARY KEY,
                    employee_name VARCHAR(150),
                    position VARCHAR(150),
                    hire_date DATE,
                    salary FLOAT,
                    department_id UUID REFERENCES departments(department_id)
                )
            """)
```

Рисунок 17. Создание таблиц департаментов и работников

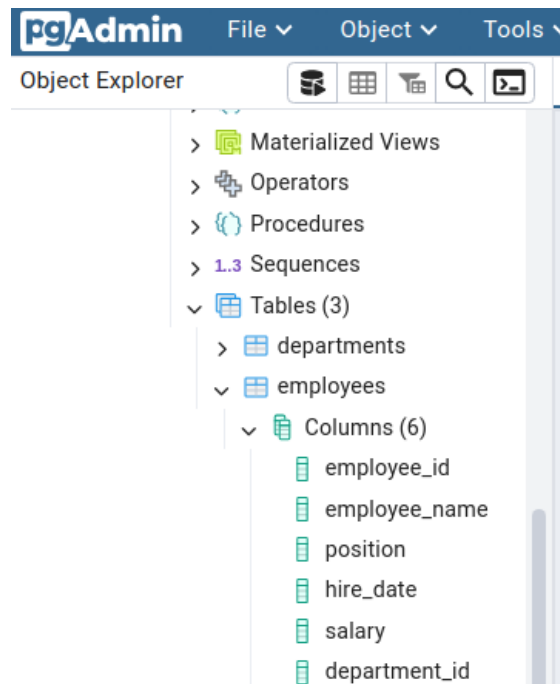


Рисунок 18. Отображение созданных таблиц в PgAdmin

Lab 4-1\_MorozovaV.ipynb

```

File Edit View Run Kernel Tabs Settings Help

+ %<  Copy Paste Run Cell All Run Code

# Загрузка данных
with pg_conn.cursor() as cur:
    start_time = datetime.datetime.now()

    departments_data = generate_department_data(n_records)
    for department in departments_data:
        cur.execute("""
            INSERT INTO departments (department_id, department_name)
            VALUES (%s, %s)
            """, (department['department_id'], department['department_name']))

    employee_data = generate_hr_data(n_records)
    for employee in employee_data:
        cur.execute("""
            INSERT INTO employees (employee_id, employee_name, position, hire_date, salary, department_id)
            VALUES (%s, %s, %s, %s, %s, %s)
            """, (employee['employee_id'], employee['employee_name'], employee['position'], employee['hire_date'],
                employee['salary'], department['department_id']))

    pg_conn.commit()
    end_time = datetime.datetime.now()
    print("Данные загружены в PostgreSQL")
    print(f"Time to load data into PostgreSQL: {end_time - start_time} seconds")

```

Рисунок 19. Загрузка генерированных данных в таблицы  
департаментов и работников

```

# Выполнение запроса MOROZOVA VALERIA
with pg_conn.cursor() as cur:
    start_time = datetime.datetime.now()
    cur.execute("""
        SELECT e.employee_name, d.department_name AS department_name, e.position
        FROM employees e
        INNER JOIN departments d ON e.department_id = d.department_id;
    """)
    results_pg = cur.fetchall()
    end_time = datetime.datetime.now()

    print("\nРезультаты анализа из PostgreSQL:")
    print(f"Time to merge data into PostgreSQL: {end_time - start_time} seconds")
    for row in results_pg:
        print(f"Имя сотрудника: {row[0]}, Название департамента: {row[1]}, Должность: {row[2]}")

except Exception as e:
    print(f"Ошибка при работе с PostgreSQL: {e}")
finally:
    pg_conn.close()
else:
    print("Пропуск операций с PostgreSQL из-за ошибки подключения")

```

Рисунок 20. Выполнение запроса по объединению таблиц

```

Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

+ 🔍 📄 ▶ ⏏ ⌂ ⏪ ⏩ Code Python 3 (ipykernel)

Успешное подключение к PostgreSQL
Данные загружены в PostgreSQL
Time to load data into PostgreSQL: 0:00:00.534567 seconds
Результаты анализа из PostgreSQL:
Time to merge data into PostgreSQL: 0:00:00.004490 seconds
Имя сотрудника: Lori Miller, Название департамента: Tourism officer, Должность: Information systems manager
Имя сотрудника: Yesenia Kennedy, Название департамента: Tourism officer, Должность: Administrator, charities/voluntary organisations
Имя сотрудника: Samuel McDowell, Название департамента: Tourism officer, Должность: Environmental education officer
Имя сотрудника: Megan Barber, Название департамента: Tourism officer, Должность: Health visitor
Имя сотрудника: Derrick Lynn, Название департамента: Tourism officer, Должность: Consulting civil engineer
Имя сотрудника: Tami Taylor, Название департамента: Tourism officer, Должность: Private music teacher

```

Рисунок 21. Результат выполнения кода

Как видно на рисунке 20, данные загружены в PostgreSQL, также время выполнения запроса по объединению значительно меньше, чем по загрузке.

```

Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

+ 🔍 📄 ▶ ⏏ ⌂ ⏪ ⏩ Code

# Группировка сотрудников по департаментам: MOROZOVA VALERIA
with pg_conn.cursor() as cur:
    start_time = datetime.datetime.now()
    cur.execute("""
        SELECT department_id, COUNT(*) FROM employees
        GROUP BY department_id;
    """)
    results_pg = cur.fetchall()
    end_time = datetime.datetime.now()

    print("\nРезультаты анализа из PostgreSQL:")
    print(f"Time to group data into PostgreSQL: {end_time - start_time} seconds")

except Exception as e:
    print(f"Ошибка при работе с PostgreSQL: {e}")
finally:
    pg_conn.close()
else:
    print("Пропуск операций с PostgreSQL из-за ошибки подключения")

Успешное подключение к PostgreSQL
Результаты анализа из PostgreSQL:
Time to group data into PostgreSQL: 0:00:00.001647 seconds

```

Рисунок 22. Результат выполнения группировки сотрудников по департаментам

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

результаты анализа из PostgreSQL:
Time to group data into PostgreSQL: 0:00:00.001647 seconds

[29]: pg_conn = check_postgres_connection(pg_conn_params)
if pg_conn:
    try:

        # Группировка сотрудников по департаментам: MOROZOVA VALERIA
        with pg_conn.cursor() as cur:
            start_time = datetime.datetime.now()
            cur.execute("""
                SELECT * FROM employees
                ORDER BY salary DESC;
            """)
            results_pg = cur.fetchall()
            end_time = datetime.datetime.now()

            print("\nРезультаты анализа из PostgreSQL:")
            print(f"Time to sort data into PostgreSQL: {end_time - start_time} seconds")

        except Exception as e:
            print(f"Ошибка при работе с PostgreSQL: {e}")
        finally:
            pg_conn.close()
    else:
        print("Пропуск операций с PostgreSQL из-за ошибки подключения")

Успешное подключение к PostgreSQL

Результаты анализа из PostgreSQL:
Time to sort data into PostgreSQL: 0:00:00.118225 seconds
```

Рисунок 23. Результат выполнения сортировки сотрудников

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

if pg_conn:
    try:

        # Сортировка сотрудников по зарплате: MOROZOVA VALERIA
        with pg_conn.cursor() as cur:
            start_time = datetime.datetime.now()
            cur.execute("""
                SELECT employee_name, position, salary FROM employees
                ORDER BY salary DESC
                LIMIT 7;
            """)
            results_pg = cur.fetchall()
            end_time = datetime.datetime.now()

            print("\nРезультаты анализа из PostgreSQL:")
            print(f"Time to sort data into PostgreSQL: {end_time - start_time} seconds")
            for row in results_pg:
                print(f"Имя сотрудника: {row[0]}, Должность: {row[1]}, Зарплата: {row[2]}")

        except Exception as e:
            print(f"Ошибка при работе с PostgreSQL: {e}")
        finally:
            pg_conn.close()
    else:
        print("Пропуск операций с PostgreSQL из-за ошибки подключения")

Успешное подключение к PostgreSQL

Результаты анализа из PostgreSQL:
Time to sort data into PostgreSQL: 0:00:00.003392 seconds
Имя сотрудника: Rebecca Torres, Должность: Research officer, political party, Зарплата: 149728.0
Имя сотрудника: Chase Lopez, Должность: Advertising art director, Зарплата: 149546.0
Имя сотрудника: Karen Smith, Должность: Early years teacher, Зарплата: 149489.0
Имя сотрудника: Cheryl Peterson, Должность: Museum/gallery conservator, Зарплата: 149433.0
Имя сотрудника: Kelsey Hale, Должность: Diplomatic Services operational officer, Зарплата: 149363.0
Имя сотрудника: Darrell Smith, Должность: Television/film/video producer, Зарплата: 149333.0
Имя сотрудника: April Thompson, Должность: Clothing/textile technologist, Зарплата: 149278.0
```

Рисунок 24. Результат выполнения сортировки сотрудников по зарплате

На рисунке 23 запрос селектит все столбцы таблицы работников и не имеет ограничений, поэтому время выполнения больше, нежели на рисунке 24, где в селекте выбирается три конкретных столбца и устанавливается лимит на вывод 7 строк. Но несмотря на это, с небольшой разницей сортировка заняла времени больше, чем группировка.

Далее похожие действия будут осуществлены в MongoDB Express, где я создала свою собственную базу и в ней коллекции (рисунок 24).

## MongoDB

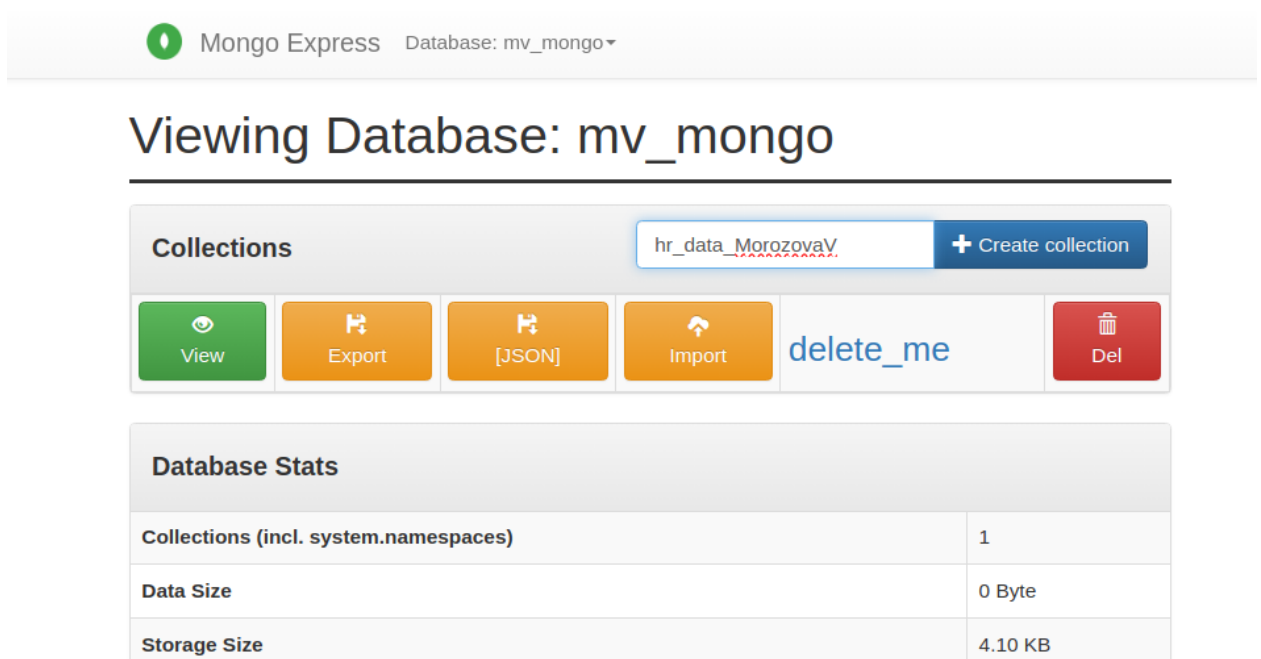


Рисунок 25. Моя база данных

```
import datetime
# Подключение к MongoDB MOROZOVA VALERIA
mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['mv_mongo']
    departments_collection = mongo_db['departments']
    employees_collection = mongo_db['employees']

# Загрузка данных в MongoDB
start_time = datetime.datetime.now()
for i in range(100):
    departments_data = generate_department_data(n_records)
    departments_collection.insert_many(departments_data)
end_time = datetime.datetime.now()
print("Данные загружены в MongoDB")
print(f"Time to load data into PostgreSQL: {end_time - start_time} seconds")
```

Успешное подключение к MongoDB

Данные загружены в MongoDB

Time to load data into PostgreSQL: 0:00:01.585682 seconds

Рисунок 26. Загрузка данных по департаментам в коллекцию

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

[23]: import datetime
# Подключение к MongoDB
mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['mv_mongo']
    departments_collection = mongo_db['departments']
    employees_collection = mongo_db['employees']

# Загрузка данных в MongoDB
start_time = datetime.datetime.now()
for i in range(100):
    employee_data = generate_hr_data(n_records)
    employees_collection.insert_many(employee_data)
end_time = datetime.datetime.now()
print("Данные загружены в MongoDB")
print(f"Time to load data into PostgreSQL: {end_time - start_time} seconds")

Успешное подключение к MongoDB
Данные загружены в MongoDB
Time to load data into PostgreSQL: 0:00:10.096796 seconds
```

Рисунок 27. Загрузка данных по работникам в коллекцию

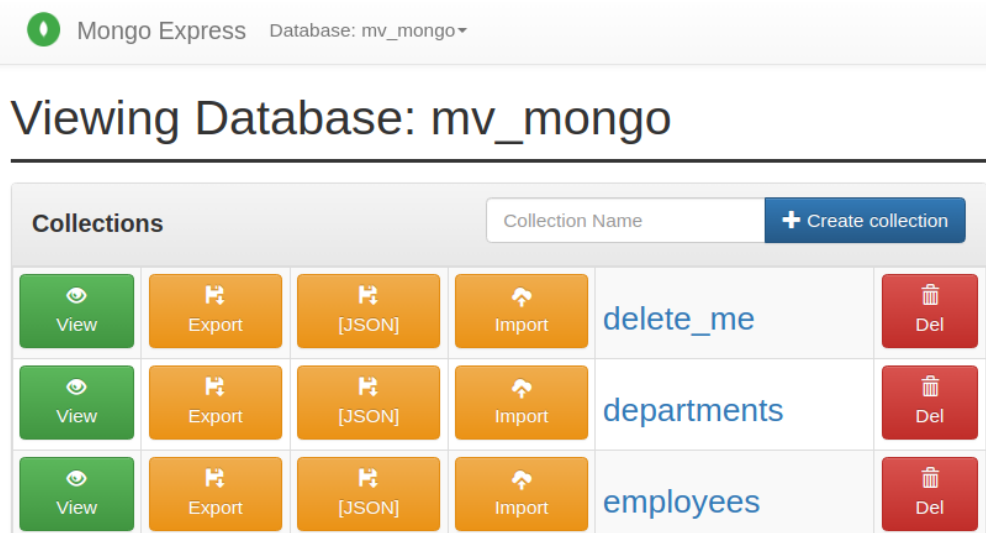


Рисунок 28. Отображение коллекций в моей базе данных

```
Lab 4-1_MorozovaV.ipynb
File Edit View Run Kernel Tabs Settings Help

[33]: import datetime
from pymongo import MongoClient

# Подключение к MongoDB
mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['mv_mongo']
    employees_collection = mongo_db['employees']
    departments_collection = mongo_db['departments']

# Пример запроса к MongoDB
start_time = datetime.datetime.now()
employee_department = employees_collection.aggregate([
    {
        "$lookup": {
            "from": "departments",
            "localField": "department_id",
            "foreignField": "id",
            "as": "department"
        }
    }
])

results_mongo = list(employee_department)
end_time = datetime.datetime.now()

print(f"Time to execute the aggregation: {end_time - start_time} seconds")
```

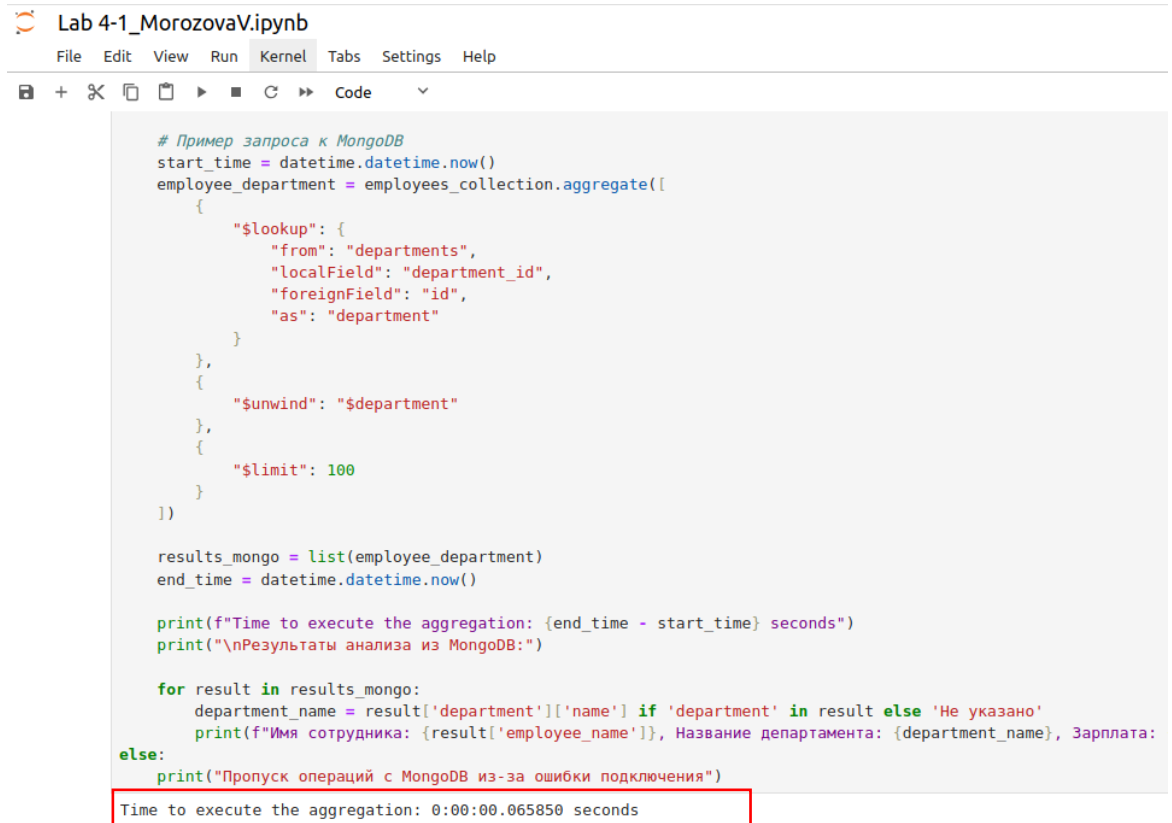
Рисунок 29. Выполнение запроса агрегации коллекций

При выполнении первичного запроса на агрегацию (рисунок 29), появилась ошибка из-за превышения лимита памяти (рисунок 30).

```
OperationFailure: PlanExecutor error during aggregation :: caused by :: Used too much memory for a single array. Memory limit: 104857600 bytes. The array contains 807861 elements and is of size 104857513 bytes. The element being added has size 132 bytes., full error: {'ok': 0.0, 'errmsg': 'PlanExecutor error during aggregation :: caused by :: Used too much memory for a single array. Memory limit: 104857600 bytes. The array contains 807861 elements and is of size 104857513 bytes. The element being added has size 132 bytes.', 'code': 146, 'codeName': 'ExceededMemoryLimit'}
```

Рисунок 30. Ошибка при агрегации

Устранить ошибку удалось с помощью изменений, внесенных в запрос, которые установили лимит (рисунок 31).



```
# Пример запроса к MongoDB
start_time = datetime.datetime.now()
employee_department = employees_collection.aggregate([
    {
        "$lookup": {
            "from": "departments",
            "localField": "department_id",
            "foreignField": "id",
            "as": "department"
        }
    },
    {
        "$unwind": "$department"
    },
    {
        "$limit": 100
    }
])

results_mongo = list(employee_department)
end_time = datetime.datetime.now()

print(f"Time to execute the aggregation: {end_time - start_time} seconds")
print("\nРезультаты анализа из MongoDB:")

for result in results_mongo:
    department_name = result['department']['name'] if 'department' in result else 'Не указано'
    print(f"Имя сотрудника: {result['employee_name']}, Название департамента: {department_name}, Зарплата: ")
else:
    print("Пропуск операций с MongoDB из-за ошибки подключения")

Time to execute the aggregation: 0:00:00.065850 seconds
```

Рисунок 31. Агрегация коллекций без ошибки

Теперь можно провести сравнительный анализ производительности и эффективности различных подходов к хранению и обработке больших данных на примере реляционной базы данных PostgreSQL и документо ориентированной базы данных MongoDB, провизуализировав результаты с помощью библиотеки matplotlib.



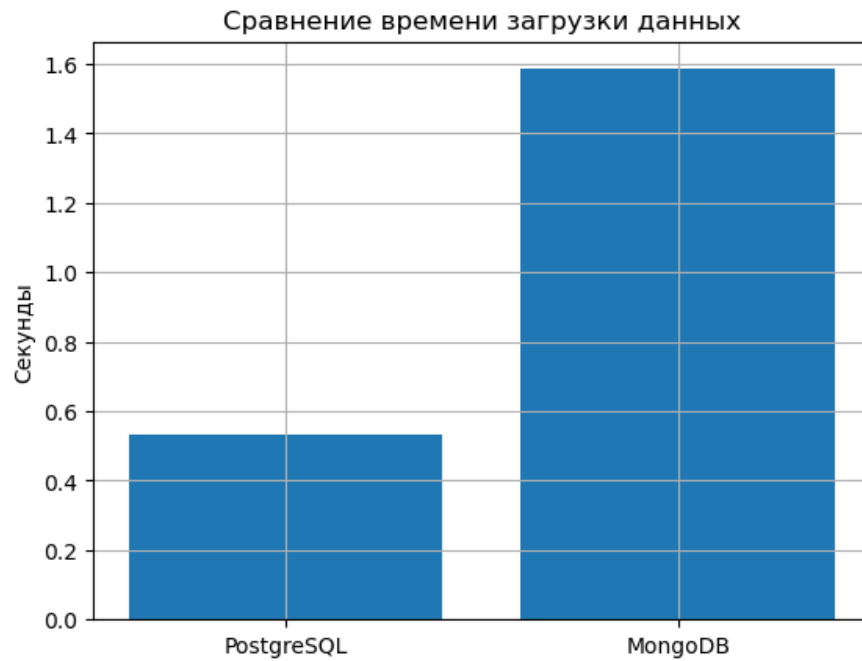


Рисунок 32. График сравнения времени на загрузку в БД

На рисунке 32 видно, что на загрузку данных в базу MongoDB ушло значительно больше времени, нежели на загрузку в базу PostgreSQL.

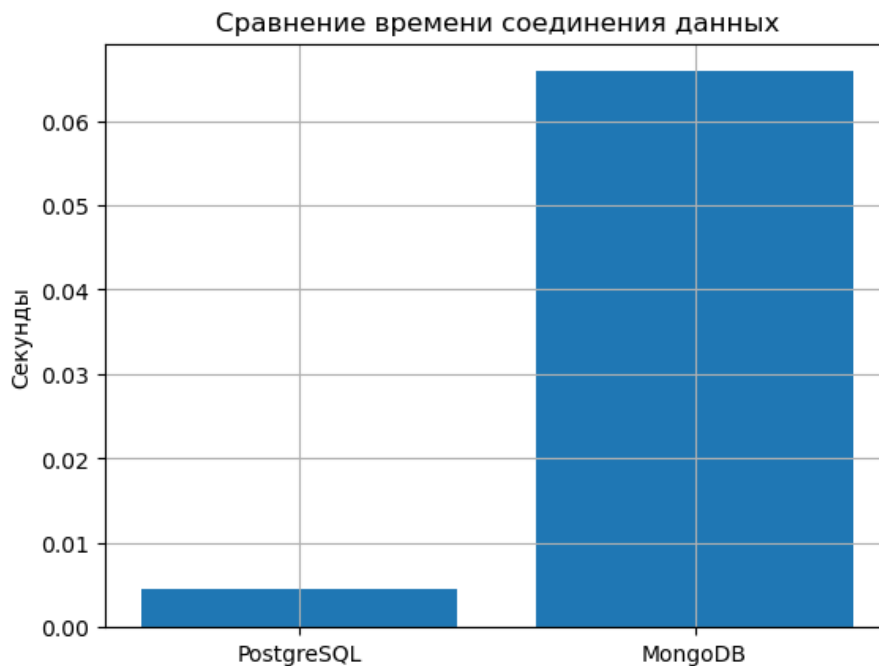


Рисунок 33. График сравнения времени на соединение данных

Рисунок 33 показывает, что на соединение (JOIN) данных в базе MongoDB ушло в разы больше времени, чем на объединение в базе PostgreSQL.

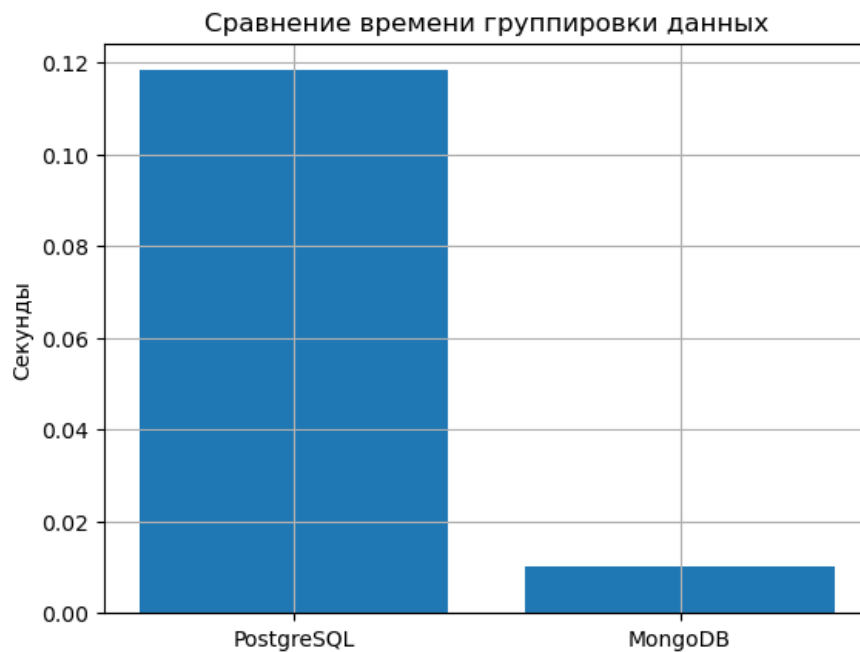


Рисунок 34. График сравнения времени на группировку данных

### Выводы

1. PostgreSQL имеет высокую производительность при записи данных, поэтому при загрузке тратит на это меньше времени.
2. PostgreSQL поддерживает JOIN, что позволяет выполнять сложные запросы данных значительно быстрее, чем MongoDB.
3. PostgreSQL требует строгой структуры данных, внесение изменений осуществляется дольше по времени.
1. MongoDB не поддерживает JOIN, что делает запросы сложнее или же вовсе выдает ошибку по причине ограничений в лимите объема данных.
2. MongoDB имеет низкую производительность при записи данных, особенно при большом объеме, в следствии чего запросы выполняются медленнее, чем в PostgreSQL.
3. MongoDB позволяет хранить данные в формате JSON, что дает возможность хранить данные в произвольной структуре и легко масштабироваться.