

Департамент образования города Москвы

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

Институт цифрового образования  
Департамент информатики, управления и технологий

**Лабораторная работа 4-1**  
**по дисциплине «Инструменты для хранения и обработки больших**  
**данных»**

**Тема: «Сравнение подходов хранения больших данных»**

Направление подготовки 38.03.05 – бизнес-информатика  
Профиль подготовки «Аналитика данных и эффективное управление»  
(очная форма обучения)

Выполнила:  
Студентка группы АДЭУ-211  
Белик Мария Константиновна

Преподаватель:  
Босенко Т.М.

Москва  
2024

## **ВВЕДЕНИЕ**

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

Оборудование и ПО:

- операционная система Ubuntu;
- MongoDB;
- PostgreSQL
- язык программирования Python;
- Библиотеки: psycopg2, pymongo, pandas, matplotlib.

## **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

В современном мире объемы данных растут экспоненциально, что приводит к необходимости использования эффективных методов их хранения и обработки. Существует два основных подхода к хранению больших данных:

1. Реляционные базы данных (например, PostgreSQL)
2. NoSQL базы данных (например, MongoDB)

Каждый из этих подходов имеет свои преимущества и недостатки, которые будут рассмотрены в ходе выполнения лабораторной работы.

## ПРАКТИЧЕСКАЯ ЧАСТЬ

### Шаг 1. Подготовка к работе

В первую очередь требуется запустить виртуальную машину, на которой установлена операционная система Ubuntu:

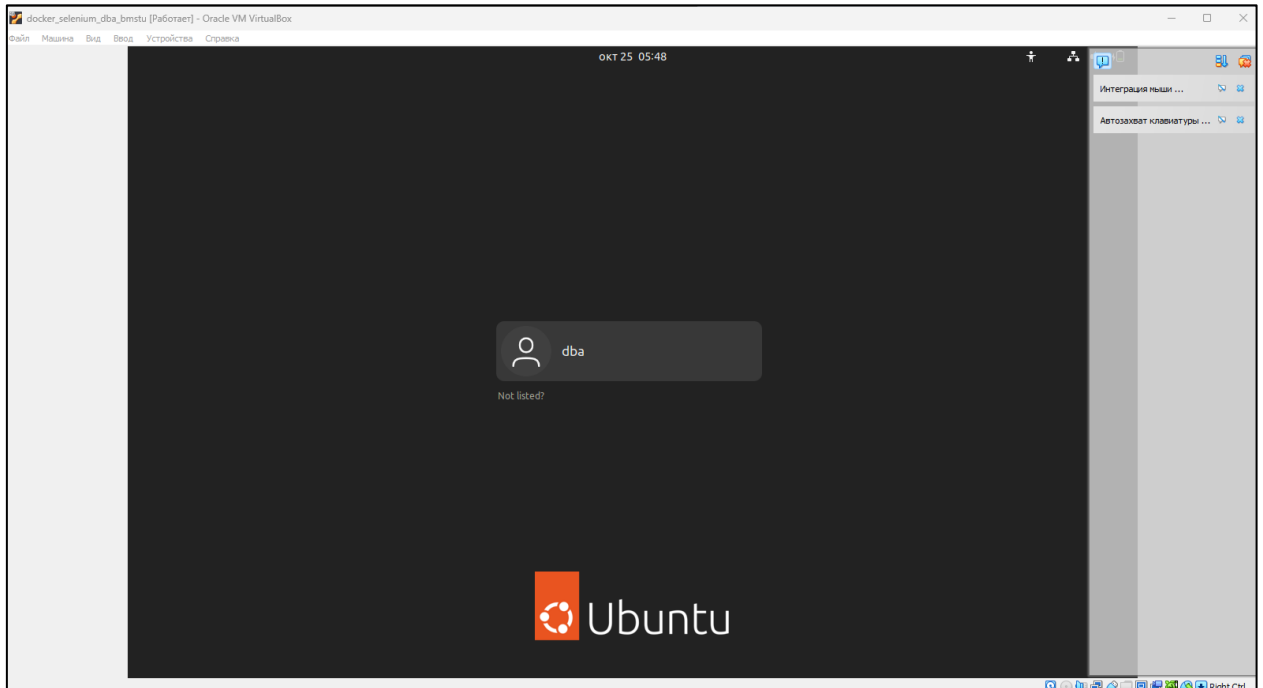


Рис. 1 – Запуск виртуальной машины

Для удобства дальнейшей работы подключимся к запущенной виртуальной машине через подключение к удаленному рабочему столу:

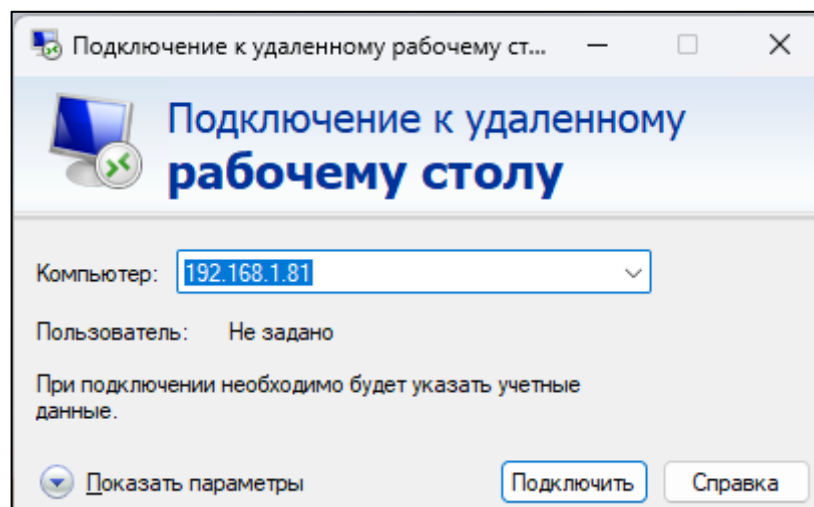


Рис. 2 – Указание настроек при подключении к удаленному рабочему столу

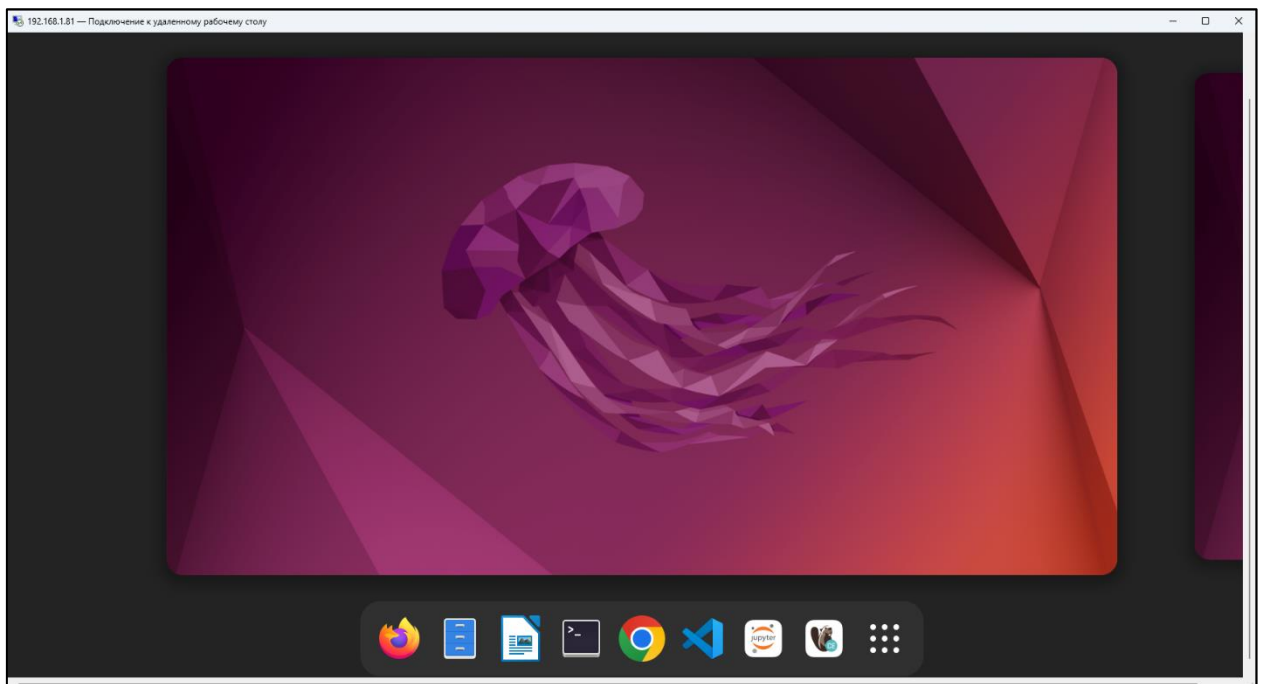


Рис. 3 – Успешное подключение к удаленному рабочему столу

Открываем файл *docker-compose.yml* в приложении “Visual Studio Code”,  
после чего запускаем докер-контейнеры командой `sudo docker compose up -d`:

```
docker-compose.yml - course_4 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  COURSE_4
    .ipynb_checkpoints
    .jupyter
    data
    postgresql
    docker-compose.yml
    lw_4.ipynb
    product_sales.csv
    ЛР_4-1.pdf
docker-compose.yml
1  services:
2    # MongoDB Service
3    mongodb:
4      container_name: mongoddb
5      image: mongo:3.5
6      environment:
7        MONGO_INITDB_ROOT_USERNAME: mongouser
8        MONGO_INITDB_ROOT_PASSWORD: mongopass
9      ports:
10       - "27017:27017"
11      volumes:
12       - mongoddb_volume:/data/db
13      networks:
14       - backend
15
16    # Mongo Express Service (Web UI for MongoDB)
17    mongo-express:
18      container_name: mongo-express
19      image: mongo-express:latest
20      environment:
21        - ME_CONFIG_MONGODB_ADMINUSERNAME=mongouser
22        - ME_CONFIG_MONGODB_ADMINPASSWORD=mongopass
23        - ME_CONFIG_BASICAUTH_USERNAME=adminuser
24        - ME_CONFIG_BASICAUTH_PASSWORD=adminpasswd
25        - ME_CONFIG_MONGODB_SERVER=mongoddb
26      ports:
27        - "8081:8081"
28      depends_on:
29        - mongodb
TERMINAL
bash
dba@dba-vm:~/Downloads/bachelor/course_4$ sudo docker compose up -d
[sudo] password for dba:
[+] Running 4/4
  ✓ Container mongoddb                Started
  ✓ Container postgres_container      Running
  ✓ Container mongo-express          Started
  ✓ Container pgadmin_container       Running
dba@dba-vm:~/Downloads/bachelor/course_4$
```

Рис. 4 – Запуск докер-контейнеров

После успешного запуска докер-контейнеров проверяем доступ к MongoDB (<http://localhost:8081>) и pgAdmin (<http://localhost:5050>) с помощью Google Chrome:

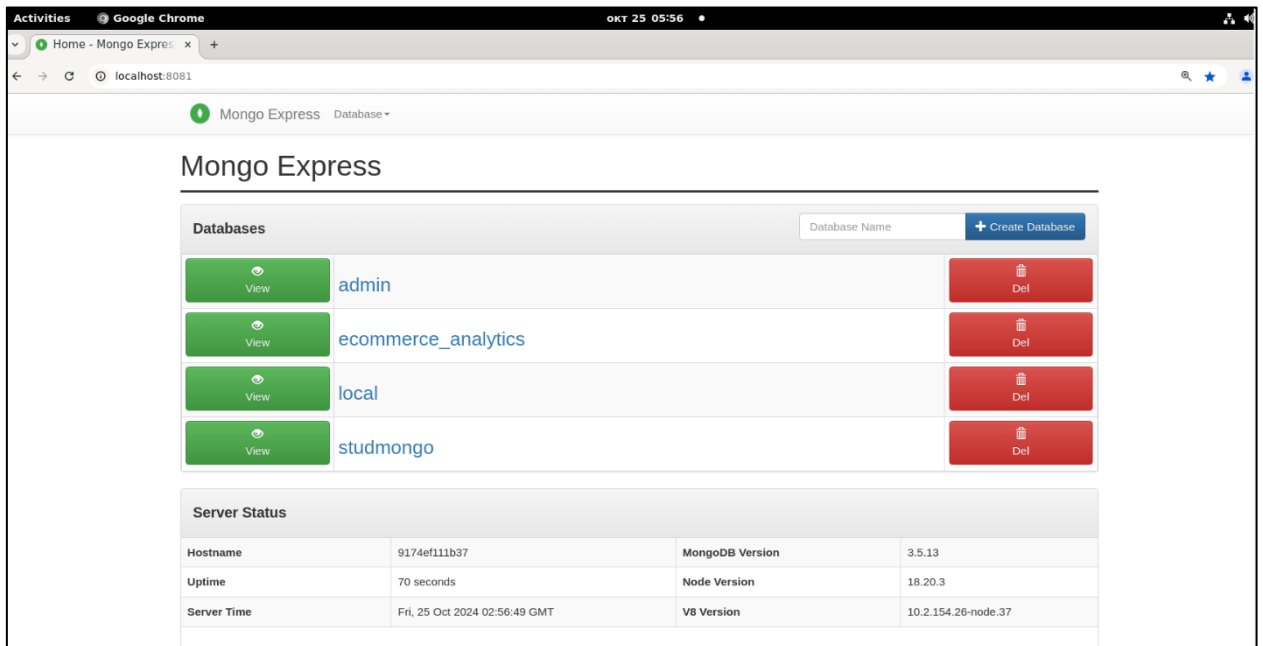


Рис. 5 – Открытие Mongo Express

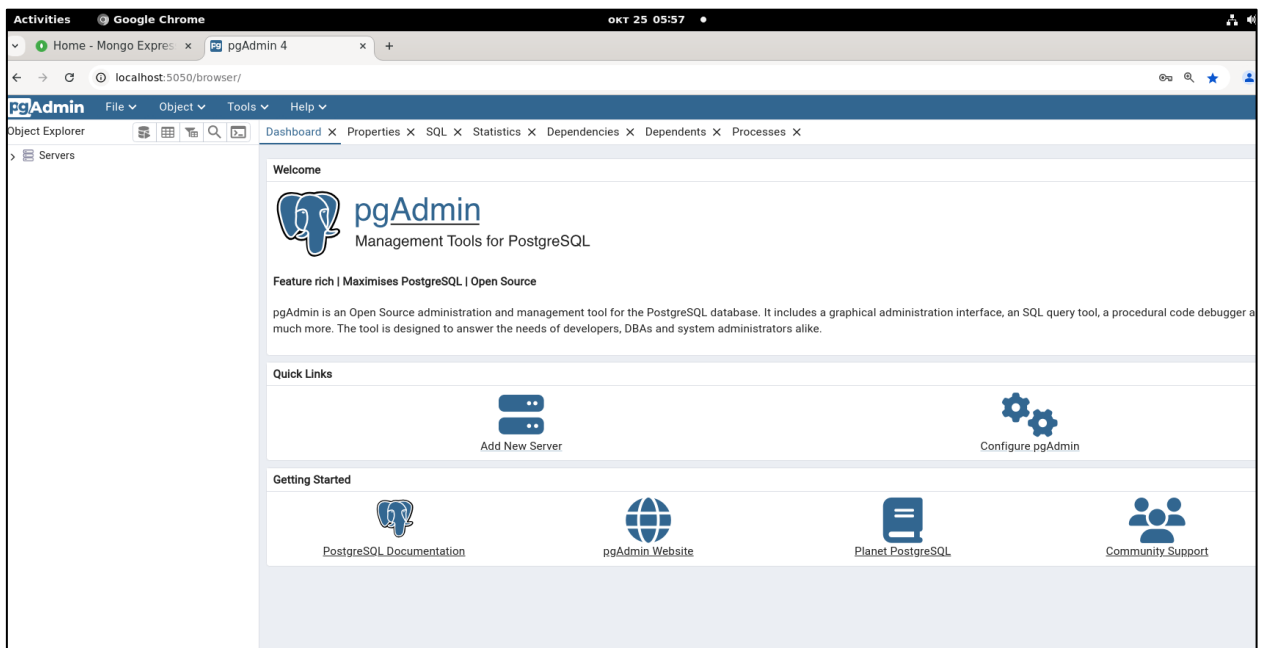
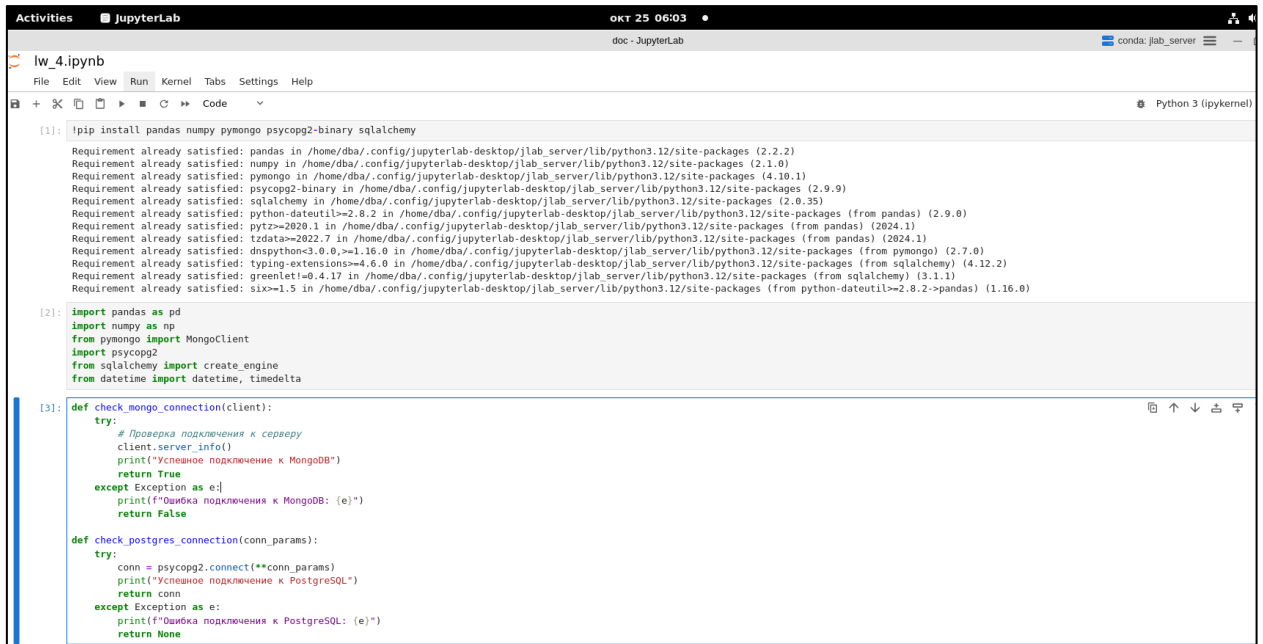


Рис. 6 – Открытие pgAdmin

## Шаг 2. Работа с MongoDB

Для того, чтобы начать работу с MongoDB необходимо создать новый блокнот в приложении “Jupyter Notebook”.

Далее требуется установить и скачать все необходимые для работы библиотеки и прописать функции для подключения к базам данных:



```
[1]: !pip install pandas numpy pymongo psycopg2-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: psycopg2-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)

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

[3]: 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
```

Рис. 7 – Скачивание необходимых для работы библиотек

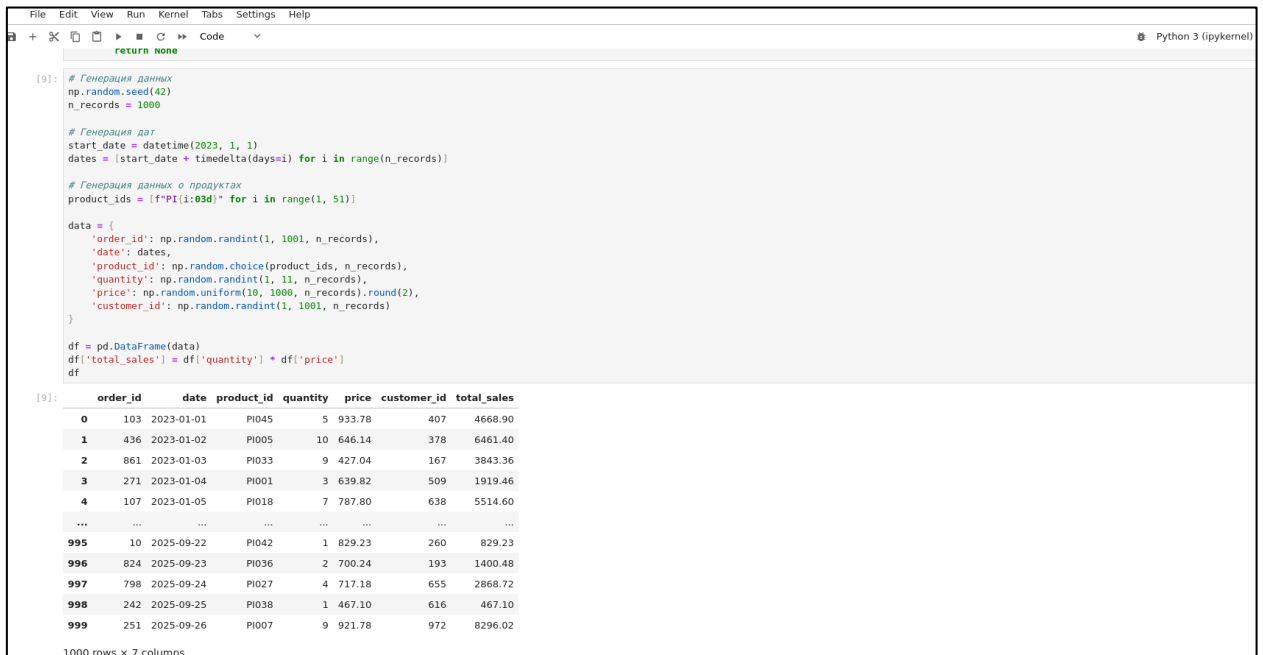
Перейдем к генерированию данных. По моему варианту (1) необходимо сгенерировать данные о продажах в интернет-магазине, согласно следующему коду примеру:

```
import random
from datetime import datetime, timedelta
import json
import psycopg2
from pymongo import MongoClient
from faker import Faker

fake = Faker()

def generate_sales_data(n_records):
    sales = []
    for _ in range(n_records):
        sale = {
            "order_id": fake.uuid4(),
            "customer_id": fake.uuid4(),
            "product_id": fake.uuid4(),
            "quantity": random.randint(1, 10),
            "price": round(random.uniform(10, 1000), 2),
            "date": fake.date_time_this_year()
        }
        sales.append(sale)
    return sales
```

Т.к. у меня возникли проблемы с Faker, то была сделана генерация требуемых данных по принципу, представленному в примере (который мы делали на лабораторном занятии):



```
[9]: # Генерация данных
np.random.seed(42)
n_records = 1000

# Генерация дат
start_date = datetime(2023, 1, 1)
dates = [start_date + timedelta(days=i) for i in range(n_records)]

# Генерация данных о продуктах
product_ids = [*PI(i:03d)* for i in range(1, 51)]

data = {
    'order_id': np.random.randint(1, 1001, n_records),
    'date': dates,
    'product_id': np.random.choice(product_ids, n_records),
    'quantity': np.random.randint(1, 11, n_records),
    'price': np.random.uniform(10, 1000, n_records).round(2),
    'customer_id': np.random.randint(1, 1001, n_records)
}

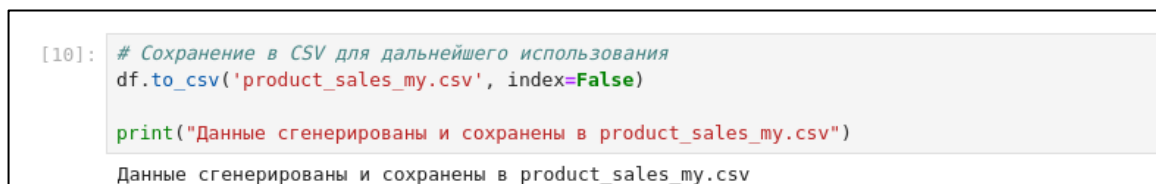
df = pd.DataFrame(data)
df['total_sales'] = df['quantity'] * df['price']
df
```

	order_id	date	product_id	quantity	price	customer_id	total_sales
0	103	2023-01-01	PI045	5	933.78	407	4668.90
1	436	2023-01-02	PI005	10	646.14	378	6461.40
2	861	2023-01-03	PI033	9	427.04	167	3843.36
3	271	2023-01-04	PI001	3	639.82	509	1919.46
4	107	2023-01-05	PI018	7	787.80	638	5514.60
...	...	...	...	...	...	...	...
995	10	2025-09-22	PI042	1	829.23	260	829.23
996	824	2025-09-23	PI036	2	700.24	193	1400.48
997	798	2025-09-24	PI027	4	717.18	655	2868.72
998	242	2025-09-25	PI038	1	467.10	616	467.10
999	251	2025-09-26	PI007	9	921.78	972	8296.02

1000 rows x 7 columns

Рис. 8 – Генерация данных о продажах интернет-магазина

Полученные данные необходимо выгрузить в файл формата csv:



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

print("Данные сгенерированы и сохранены в product_sales_my.csv")
```

Данные сгенерированы и сохранены в product\_sales\_my.csv

Рис. 9 – Выгрузка данных в файл «product\_sales\_my.csv»

Полученные данные о продажах интернет-магазина нужно загрузить в базу данных MongoDB, для этого подключаемся к MongoDB, загружаем данные, проверяем успешность выполнения действий, делаем запрос к базе данных, результатом которого является вывод id товара, общей суммы, на которую было продано данного товара, и средней цены данного товара с фильтрацией по общей сумме (от наибольшей к наименьшей):

```
File Edit View Run Kernel Tabs Settings Help
Python 3 (ipykernel)

[12]: # Подключение к MongoDB
mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['studmongo']
    collection = mongo_db['my_sales']

    # Загрузка данных в MongoDB
    records = df.to_dict('records')
    collection.insert_many(records)
    print("Данные загружены в MongoDB")

    # Пример запроса к MongoDB
    pipeline = [
        {"$group": {
            "_id": "$product_id",
            "total_sales": {"$sum": "$total_sales"},
            "avg_price": {"$avg": "$price"}
        }},
        {"$sort": {"total_sales": -1}}
    ]

    results_mongo = list(collection.aggregate(pipeline))
    print("\nРезультаты анализа из MongoDB:")
    for result in results_mongo:
        print(f"Product id: {result['_id']}, Общие продажи: {result['total_sales']:.2f}, Средняя цена: {result['avg_price']:.2f}")

else:
    print("Пропуск операций с MongoDB из-за ошибки подключения")

Успешное подключение к MongoDB
Данные загружены в MongoDB

Результаты анализа из MongoDB:
Product id: PI026, Общие продажи: 96072.99, Средняя цена: 508.69
Product id: PI033, Общие продажи: 85648.54, Средняя цена: 668.67
Product id: PI040, Общие продажи: 84120.90, Средняя цена: 518.98
Product id: PI047, Общие продажи: 82621.67, Средняя цена: 471.40
Product id: PI025, Общие продажи: 77754.70, Средняя цена: 459.86
Product id: PI011, Общие продажи: 74051.32, Средняя цена: 609.31
Product id: PI049, Общие продажи: 73202.07, Средняя цена: 524.61
Product id: PI034, Общие продажи: 72799.13, Средняя цена: 494.63
Product id: PI009, Общие продажи: 72753.25, Средняя цена: 630.61
Product id: PI006, Общие продажи: 71573.87, Средняя цена: 558.98
Product id: PI003, Общие продажи: 69801.01, Средняя цена: 581.58
Product id: PI037, Общие продажи: 67013.29, Средняя цена: 514.93
```

Рис. 10 – Подключение к MongoDB, загрузка данных и выполнение запроса



```

Результаты анализа из MongoDB:
Product id: PI026, Общие продажи: 96072.99, Средняя цена: 508.69
Product id: PI033, Общие продажи: 85648.54, Средняя цена: 660.67
Product id: PI040, Общие продажи: 84120.90, Средняя цена: 518.98
Product id: PI047, Общие продажи: 82621.67, Средняя цена: 471.40
Product id: PI025, Общие продажи: 77754.70, Средняя цена: 459.86
Product id: PI011, Общие продажи: 74051.32, Средняя цена: 609.31
Product id: PI049, Общие продажи: 73202.07, Средняя цена: 524.61
Product id: PI034, Общие продажи: 72799.13, Средняя цена: 494.63
Product id: PI009, Общие продажи: 72753.25, Средняя цена: 630.61
Product id: PI006, Общие продажи: 71573.87, Средняя цена: 558.98
Product id: PI003, Общие продажи: 69801.81, Средняя цена: 581.58
Product id: PI037, Общие продажи: 67013.29, Средняя цена: 514.93
Product id: PI017, Общие продажи: 65222.02, Средняя цена: 466.79
Product id: PI010, Общие продажи: 63997.55, Средняя цена: 567.39
Product id: PI038, Общие продажи: 63820.31, Средняя цена: 618.53
Product id: PI007, Общие продажи: 62536.38, Средняя цена: 592.59
Product id: PI028, Общие продажи: 61872.78, Средняя цена: 642.16
Product id: PI024, Общие продажи: 61614.48, Средняя цена: 528.87
Product id: PI044, Общие продажи: 61104.00, Средняя цена: 587.07
Product id: PI023, Общие продажи: 60271.37, Средняя цена: 448.19
Product id: PI045, Общие продажи: 59513.39, Средняя цена: 494.98
Product id: PI004, Общие продажи: 59229.62, Средняя цена: 543.33
Product id: PI032, Общие продажи: 58054.36, Средняя цена: 641.12
Product id: PI029, Общие продажи: 57231.10, Средняя цена: 430.11
Product id: PI005, Общие продажи: 57092.18, Средняя цена: 421.24
Product id: PI030, Общие продажи: 56775.61, Средняя цена: 431.71
Product id: PI046, Общие продажи: 55804.31, Средняя цена: 611.88
Product id: PI018, Общие продажи: 54927.70, Средняя цена: 493.72
Product id: PI035, Общие продажи: 53326.43, Средняя цена: 438.01
Product id: PI014, Общие продажи: 52875.72, Средняя цена: 500.97
Product id: PI031, Общие продажи: 51385.47, Средняя цена: 549.68
Product id: PI050, Общие продажи: 49361.03, Средняя цена: 656.59
Product id: PI039, Общие продажи: 48025.57, Средняя цена: 430.05
Product id: PI002, Общие продажи: 47449.45, Средняя цена: 494.75
Product id: PI021, Общие продажи: 46597.93, Средняя цена: 444.66
Product id: PI013, Общие продажи: 44668.65, Средняя цена: 486.27
Product id: PI008, Общие продажи: 43903.38, Средняя цена: 453.69
Product id: PI043, Общие продажи: 40147.09, Средняя цена: 390.69
Product id: PI022, Общие продажи: 39565.28, Средняя цена: 502.49
Product id: PI027, Общие продажи: 38533.33, Средняя цена: 457.90
Product id: PI012, Общие продажи: 38073.97, Средняя цена: 548.82

```

Рис. 11 – Результат анализа из MongoDB

### Шаг 3. Работа с PostgreSQL

Первым этапом работы аналогично является подключение к PostgreSQL:

```
[13]: import psycopg2

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

Подключение успешно

[16]: # 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:
        # Создание таблицы
        with pg_conn.cursor() as cur:
            cur.execute("""
                CREATE TABLE IF NOT EXISTS my_sales_2 (
                    order_id VARCHAR(10),
                    date DATE,
                    product_id VARCHAR(10),
                    quantity INTEGER,
                    price FLOAT,
                    customer_id INTEGER,
                    total_sales FLOAT
                )
            """)
    except Exception as e:
        print(f"Ошибка при создании таблицы: {e}")
    else:
        print("Таблица my_sales_2 успешно создана")
```

Рис. 12 – Подключение к PostgreSQL

Далее загрузим сгенерированные данные о продажах в интернет-магазине и выполним запрос, результатом которого является вывод id товара, общей суммы, на которую было продано данного товара, и средней цены данного товара с фильтрацией по id товара:

```
# Загрузка данных
with pg_conn.cursor() as cur:
    for _, row in df.iterrows():
        cur.execute("""
            INSERT INTO my_sales_2 (order_id, date, product_id, quantity, price, customer_id, total_sales)
            VALUES (%s, %s, %s, %s, %s, %s, %s)
        """, (row['order_id'], row['date'], row['product_id'], row['quantity'], row['price'],
            row['customer_id'], row['total_sales']))

pg_conn.commit()
print("Данные загружены в PostgreSQL")

# Выполнение запроса
with pg_conn.cursor() as cur:
    cur.execute("""
        SELECT product_id,
            SUM(total_sales) as total_sales,
            AVG(price) as avg_price
        FROM my_sales_2
        GROUP BY product_id
        ORDER BY product_id
    """)
    results_pg = cur.fetchall()

print("\nРезультаты анализа из PostgreSQL:")
for row in results_pg:
    print(f"Product id: {row[0]}, Общие продажи: {row[1]:.2f}, Средняя цена: {row[2]:.2f}")

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
Данные загружены в PostgreSQL

Результаты анализа из PostgreSQL:
Product id: P1001, Общие продажи: 37885.20, Средняя цена: 379.75
Product id: P1002, Общие продажи: 47449.45, Средняя цена: 494.75
Product id: P1003, Общие продажи: 69801.81, Средняя цена: 581.58
```

Рис. 13 – Загрузка данных в PostgreSQL, выполнение запроса

Результаты анализа из PostgreSQL:

Product id:	PI001,	Общие продажи:	37885.20,	Средняя цена:	379.75
Product id:	PI002,	Общие продажи:	47449.45,	Средняя цена:	494.75
Product id:	PI003,	Общие продажи:	69801.81,	Средняя цена:	581.58
Product id:	PI004,	Общие продажи:	59229.62,	Средняя цена:	543.33
Product id:	PI005,	Общие продажи:	57092.18,	Средняя цена:	421.24
Product id:	PI006,	Общие продажи:	71573.87,	Средняя цена:	558.98
Product id:	PI007,	Общие продажи:	62536.38,	Средняя цена:	592.59
Product id:	PI008,	Общие продажи:	43903.38,	Средняя цена:	453.69
Product id:	PI009,	Общие продажи:	72753.25,	Средняя цена:	630.61
Product id:	PI010,	Общие продажи:	63997.55,	Средняя цена:	567.39
Product id:	PI011,	Общие продажи:	74051.32,	Средняя цена:	609.31
Product id:	PI012,	Общие продажи:	38073.97,	Средняя цена:	548.82
Product id:	PI013,	Общие продажи:	44668.65,	Средняя цена:	486.27
Product id:	PI014,	Общие продажи:	52875.72,	Средняя цена:	500.97
Product id:	PI015,	Общие продажи:	33512.94,	Средняя цена:	420.72
Product id:	PI016,	Общие продажи:	30801.55,	Средняя цена:	518.80
Product id:	PI017,	Общие продажи:	65222.02,	Средняя цена:	466.79
Product id:	PI018,	Общие продажи:	54927.70,	Средняя цена:	493.72
Product id:	PI019,	Общие продажи:	20815.68,	Средняя цена:	331.43
Product id:	PI020,	Общие продажи:	25101.88,	Средняя цена:	354.03
Product id:	PI021,	Общие продажи:	46597.93,	Средняя цена:	444.66
Product id:	PI022,	Общие продажи:	39565.28,	Средняя цена:	502.49
Product id:	PI023,	Общие продажи:	60271.37,	Средняя цена:	448.19
Product id:	PI024,	Общие продажи:	61614.48,	Средняя цена:	528.87
Product id:	PI025,	Общие продажи:	77754.70,	Средняя цена:	459.86
Product id:	PI026,	Общие продажи:	96072.99,	Средняя цена:	508.69
Product id:	PI027,	Общие продажи:	38533.33,	Средняя цена:	457.90
Product id:	PI028,	Общие продажи:	61872.78,	Средняя цена:	642.17
Product id:	PI029,	Общие продажи:	57231.10,	Средняя цена:	430.11
Product id:	PI030,	Общие продажи:	56775.61,	Средняя цена:	431.71
Product id:	PI031,	Общие продажи:	51385.47,	Средняя цена:	549.68
Product id:	PI032,	Общие продажи:	58054.36,	Средняя цена:	641.12
Product id:	PI033,	Общие продажи:	85648.54,	Средняя цена:	660.67
Product id:	PI034,	Общие продажи:	72799.13,	Средняя цена:	494.63
Product id:	PI035,	Общие продажи:	53326.43,	Средняя цена:	438.01
Product id:	PI036,	Общие продажи:	34902.46,	Средняя цена:	449.60
Product id:	PI037,	Общие продажи:	67013.29,	Средняя цена:	514.93
Product id:	PI038,	Общие продажи:	63820.31,	Средняя цена:	618.53
Product id:	PI039,	Общие продажи:	48025.57,	Средняя цена:	430.05
Product id:	PI040,	Общие продажи:	84120.90,	Средняя цена:	518.98
Product id:	PI041,	Общие продажи:	30157.90,	Средняя цена:	491.07
Product id:	PI042,	Общие продажи:	33139.08,	Средняя цена:	499.44
Product id:	PI043,	Общие продажи:	40147.09,	Средняя цена:	390.69
Product id:	PI044,	Общие продажи:	61104.00,	Средняя цена:	587.07
Product id:	PI045,	Общие продажи:	59513.39,	Средняя цена:	494.98

Рис. 14 – Результат анализа из PostgreSQL

## ВЫВОДЫ

Таким образом, в ходе выполнения данной лабораторной работы:

- был отработан DevOps подход (развертывание необходимого ПО с помощью докер-контейнеров);
- была продемонстрирована работа с реляционной базой данных PostgreSQL;
- была продемонстрирована работа с базой данных NoSQL – MongoDB.

Также хочется отметить преимущества NoSQL базы данных над реляционной, т.к. все-таки она позволяет добавлять объекты нового типа без изменения схемы базы данных, позволяет работать с различными типами данных (не встраиваемыми в одну структуру и собираемыми из разных источников), а так же позволяет обрабатывать сложные запросы быстрее.