САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Тетюхин Максим, Лукьянов Платон, Котегов Никита, Пан Владимир

Практика использования триггеров для БД с научными журналами и статьями

Задание в команде

Преподаватели:

Михайлов Дмитрий Андреевич

Шевнин Лев Ярославович

Санкт-Петербург

2025

1. Выбор сайта и парсинг данных

Выбор сайта для подбора датасета остановился на сайте «КиберЛенинка» (https://cyberleninka.ru/). Данный сайт представляет из себя большую онлайн библиотеку научных работ и статей.

Через **Network Inspector** найден внутренний API-запрос CyberLeninka, по которому получен полный список статей. Ответ сверстан в единый JSON-файл. Далее реализован парсер карточки статьи на основе **BeautifulSoup**, извлекающий просмотры, скачивания, ключевые слова и прочие метаданные. Для статей с DOI выполнен дополнительный запрос к API **CrossRef** — получены дата индексации и показатели цитирования. Аналогичным образом разработан парсер страницы журнала; сведения о просмотрах, скачиваниях, индексе Хирша и каталогах сохранены во второй JSON-файл.


```
{
  "articles": [
    {
      "views": 39,
      "downloads": 5,
      "doi": "10.24412/2220-7880-2024-3-60-66",
      "indexed_datetime": null,
      "reference_count": null,
      "is_referenced_by_count": null,
      "volume": null,

      "year": 2024,
      "name": "ОЦЕНКА ВАЛИДНОСТИ …",
      "annotation": "Течение большинства психических заболеваний …",
      "link": "/article/n/otsenka-validnosti-i-nadezhnosti-…",

      "journal": "Вятский медицинский вестник",
      "journal_link": "/journal/n/vyatskiy-meditsinskiy-vestnik",

      "research_area": "Медицинские науки и общественное здравоохранение",

      "authors": [
        "Зыкова А. С.",
        "Оправин А. С."
      ],

      "key_words": [
        "комплаентность",
        "психические заболевания",
        "compliance",
        "mental diseases"
      ],

      "ocr": [
        "Фрагмент распознанного текста №1 …",
        "Фрагмент распознанного текста №2 …"
      ],

      "catalogs": [
        {
          "id": 8,
          "alias_name": "vak",
          "acronym": "ВАК"
        }
      ]
    }
    /* … другие статьи … */
  ]
}
```


Рисунок 1 - образец по которому составлен articles.json

```
{
  "journals": [
    {
      "name": "Вестник Чувашского государственного педагогического университета им. И. Я. Яковлева",
      "link": "/journal/n/vestnik-chuvashskogo-gosudarstvennogo-pedagogicheskogo-universiteta-im-i-ya-yakovleva",

      "views": 1195499,
      "downloads": 129812,
      "hirsh_index": 9,
      "articles_count": 2156,

      "catalogs": [
        "ВАК"
      ]
    },
    {
      "name": "Сибирский психологический журнал",
      "link": "/journal/n/sibirskiy-psihologicheskiy-zhurnal",

      "views": 1256528,
      "downloads": 166182,
      "hirsh_index": 11,
      "articles_count": 1315,

      "catalogs": [
        "Scopus",
        "ВАК",
        "ESCI"
      ]
    }
    /* … другие журналы … */
  ]
}
```


Рисунок 2 - образец по которому сделан journals.json

2. Создание таблицы

JSON-дампы *journals.json* и *full_articles.json* прочитаны в объекты DataFrame библиотеки pandas. Далее нормализовали журналы
Далее выделили базовые сущности:

1. *Журнал* (*journals*) — агрегированная статистика из карточки журнала.
2. *Статья* (*articles*) — все метаданные одной публикации, включая ссылки на журнал и DOI-поля.
3. Справочники-«словарики»:
    a. *catalogs* — типы индексации («ВАК», *Scopus*…).
    b. *authors* — уникальные ФИО.
    c. *key_words* — нормализованные ключевые слова.

Далее выгрузили данные в PostrgreSQL.

3. Анализ данных

Для анализа данных использовали библиотеки *pandas*, *matplotlib.pyplot*.
В рамках первичного анализа данных были построены следующие зависимости:

1. **Гистограмма просмотров журналов**
   Цель: оценить распределение количества просмотров на уровне журналов.
   Вывод: сильная асимметрия вправо; большинство журналов имеют до 200 тыс. просмотров, но есть единичные с миллионами.

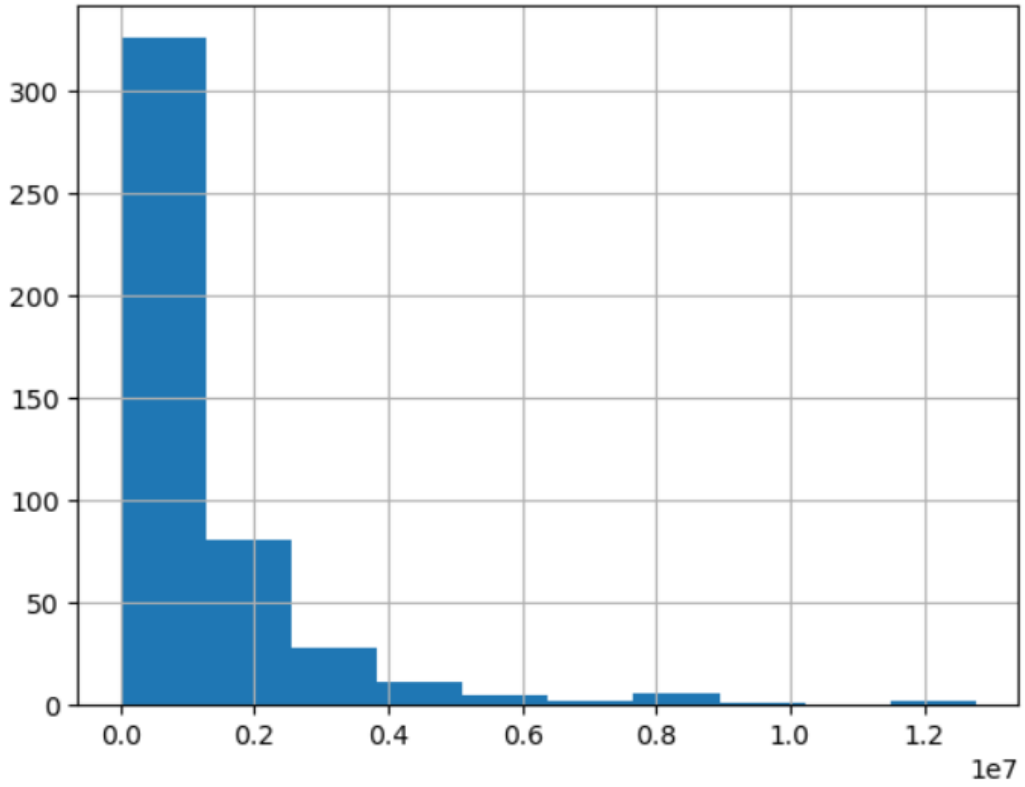

Рисунок 3 - зависимость какому количеству журналов соответствует какое количество просмотров

2. **Гистограмма скачиваний журналов**
   Цель: проанализировать общее количество скачиваний для каждого журнала.
   Вывод: форма распределения аналогична просмотрам, но значения на 1–2 порядка меньше; скачивания прямо пропорциональны просмотрам.

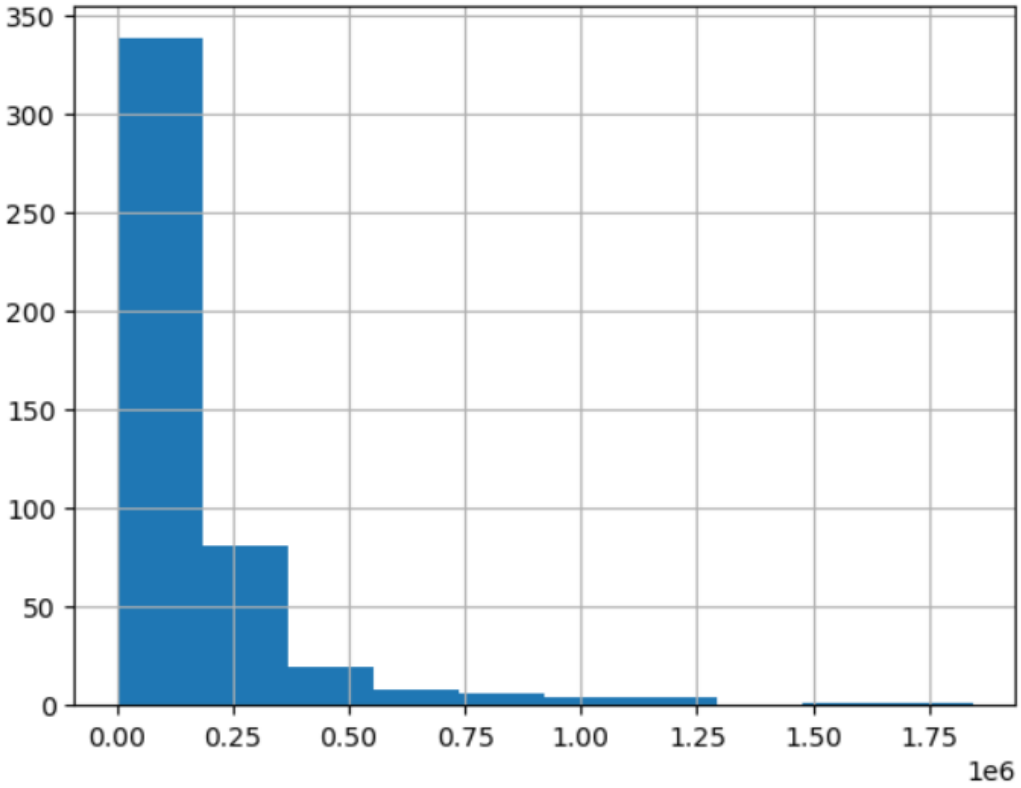

Рисунок 4 - зависимость какому количеству журналов соответствует какое количество скачиваний

3. **Гистограмма просмотров статей**
   Цель: изучить популярность отдельных статей по просмотрам.
   Вывод: у большинства статей — менее 100 просмотров, максимум — до ~2000; длинный хвост менее выражен, чем у журналов.

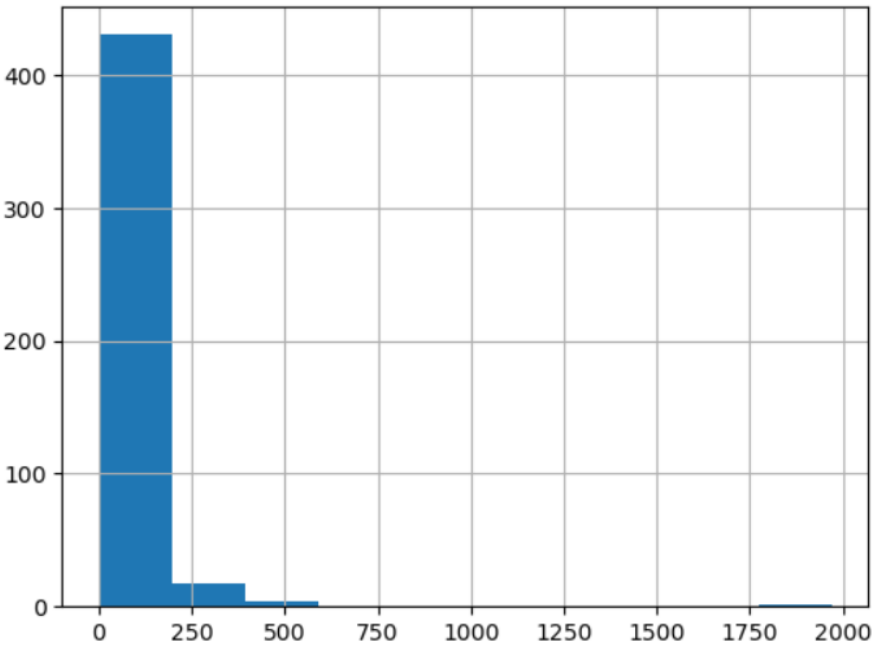

Рисунок 5 - зависимость какому количеству научных статей соответствует какое количество просмотров

4. **Гистограмма скачиваний статей**
   Цель: понять уровень интереса к статьям в формате загрузки.
   Вывод: чаще всего статьи скачивают 1–2 раза; форма совпадает с графиком просмотров, но значения ещё ниже.

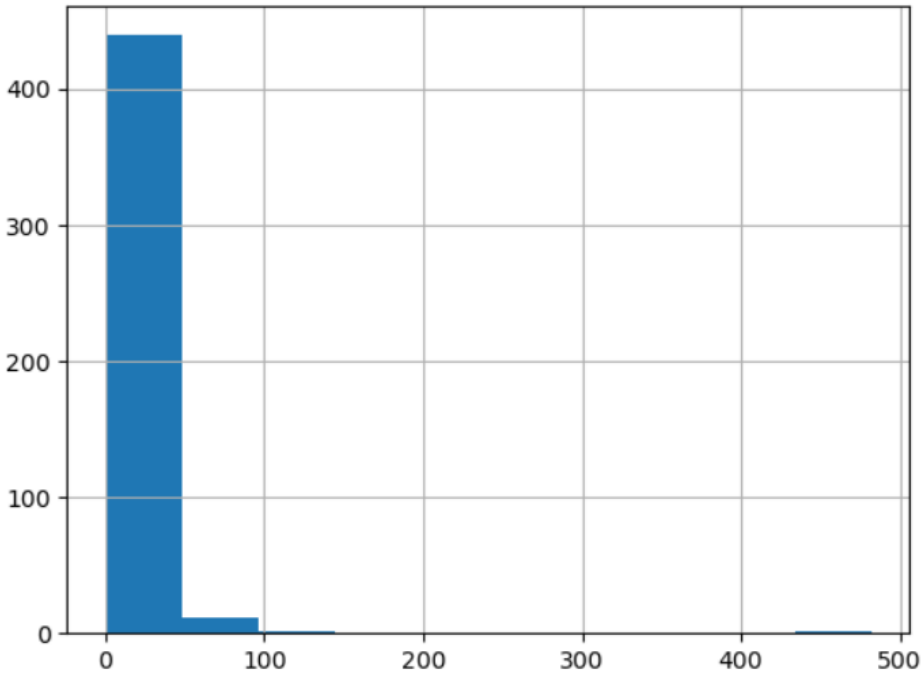

Рисунок 6 - зависимость какому количеству научных статей соответствует какое количество скачиваний

5. Другие данные:

```
print(df_1['downloads'].mean()) #Среднее для downloads
print(df_1['downloads'].std()) #Станд отклонение для downloads
print(df_1['downloads'].median()) #Медиана для downloads
print('------------------')
print(df_1['views'].mean()) #Станд отклонение для views
print(df_1['views'].std()) #Станд отклонение для views
print(df_1['views'].median()) #Медиана для downloads
```

```
164566.65367965368
223174.08832389794
97074.0
------------------
1237482.2121212122
1628653.5975722428
770250.5
```

```
print(df_2['downloads'].mean()) #Среднее для downloads
print(df_2['downloads'].std()) #Станд отклонение для downloads
print(df_2['downloads'].median()) #Станд отклонение для downloads
print('------------------')
print(df_2['views'].mean()) #Станд отклонение для views
print(df_2['views'].std()) #Станд отклонение для views
print(df_2['views'].median()) #Станд отклонение для views
```

```
11.391592920353983
25.992822139553958
6.0
------------------
61.075221238938056
111.71212848534844
37.0
```

4. Добавление триггеров на базовые запросы в БД

```sql
CREATE TABLE journal_audit_log ( -- Таблица для логов
    log_id SERIAL PRIMARY KEY,
    operation VARCHAR(10),
    journal_name TEXT,
    changed_at TIMESTAMP DEFAULT NOW(),
    old_data JSONB,
    new_data JSONB
);
```

- Создаётся таблица логов: journal_audit_log
    - Здесь будет храниться **каждое изменение** журнала:
      – какая операция была сделана,
      – над каким журналом,
      – что именно изменилось (в формате JSON).

```sql
CREATE OR REPLACE FUNCTION log_journal_changes() -- Функция для логгирования
RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN -- Удаление
        INSERT INTO journal_audit_log (operation, journal_name, old_data)
        VALUES ('DELETE', OLD.name, to_jsonb(OLD));
    ELSIF (TG_OP = 'UPDATE') THEN -- Изменение
        INSERT INTO journal_audit_log (operation, journal_name, old_data, new_data)
        VALUES ('UPDATE', NEW.name, to_jsonb(OLD), to_jsonb(NEW));
    ELSIF (TG_OP = 'INSERT') THEN -- Вставка
        INSERT INTO journal_audit_log (operation, journal_name, new_data)
        VALUES ('INSERT', NEW.name, to_jsonb(NEW));
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

- Создаётся функция-триггер: log_journal_changes()
    - Эта функция вызывается автоматически перед каждой вставкой, обновлением или удалением строки из таблицы journals.
    - Она записывает:
      INSERT → только new_data
      UPDATE → old_data и new_data
      DELETE → только old_data

```sql
CREATE TRIGGER journal_audit_trigger
AFTER INSERT OR UPDATE OR DELETE ON journals
FOR EACH ROW
EXECUTE FUNCTION log_journal_changes();
```

- Этот триггер **связывает** таблицу journals с функцией.
  Теперь при любом изменении (добавлении, удалении, редактировании журнала) будет автоматически сохраняться запись в journal_audit_log.

5. Добавление реализации индексов в БД

```
CREATE INDEX link_ind ON articles(link);
CREATE INDEX author ON articles(link);
CREATE INDEX ind_doi ON articles(doi DESC);
CREATE INDEX ind_views ON articles(views) WHERE views>100;
```

- Индекс по полю *link*, которое используется как первичный идентификатор статьи и активно применяется при соединениях (JOIN) с другими таблицами (article_author, article_keyword). Повышает производительность при точечном поиске статьи по ссылке
- Этот индекс создаётся по полю *link* в таблице *articles*. Он используется для ускорения операций соединения (JOIN) между таблицей articles и вспомогательными таблицами, такими как *article_author* или *article_keyword*, где *link* служит внешним ключом
- Индекс по полю *doi* с сортировкой по убыванию. Это полезно для запросов, где требуется быстро получить статьи с наибольшим значением *DOI*
- Частичный (или фильтрованный) индекс — применяется только к статьям, у которых просмотров больше 100. Это позволяет оптимизировать запросы