



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра прикладной математики (ПМ)

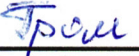
КУРСОВАЯ РАБОТА

по дисциплине

«Системы управления данными»

Тема курсовой работы: «Сбор, предобработка и анализ данных о пользователях группы «Кафедра прикладной математики»»

Студент группы ИМБО-02-21 Громов Павел Сергеевич


(подпись)

Руководитель
курсовой работы

ст. преподаватель Буданцев А.В.


(подпись)

Работа представлена к защите «__» _____ 2024 г.

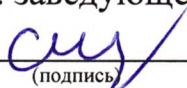
Допущен к защите «__» _____ 2024 г.

Москва 2024 г.



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра прикладной математики (ПМ)

Утверждаю
и.о. заведующего кафедрой ПМ
 Смоленцева Т.Е.
(подпись) « » 2024 г.

ЗАДАНИЕ
на выполнение курсовой работы
по дисциплине «Системы управления данными»

Студент Громов Павел Сергеевич

Группа ИМБО-02-21

Тема «Сбор, предобработка и анализ данных о пользователях группы «Кафедра прикладной математики»»

Исходные данные: Исходные данные о взяты из группы «Кафедра прикладной математики»

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

Вывести 10 самых активных пользователей.

Провести корреляционный анализ активности пользователя в группе и на личной странице.

Провести корреляционный анализ характеристик постов в группе.

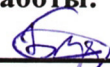
Провести кластеризацию постов по характеристикам.

Построить графики распределения полов в группе и активности по полу.

Срок представления к защите курсовой работы:

до « » 2024 г.

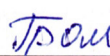
Задание на курсовую работу выдал


Подпись руководителя

Буданцев А.В.
(ФИО руководителя)

« » 2024 г.

Задание на курсовую работу получил


Подпись обучающегося


(ФИО обучающегося)

« » 2024 г.

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	4
ВВЕДЕНИЕ	5
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	6
1.1 ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ИНСТРУМЕНТОВ	6
2 ПРАКТИЧЕСКАЯ ЧАСТЬ	8
2.1 УСТАНОВКА, КОНФИГУРАЦИЯ И ПРИМЕНЕНИЕ ИНСТРУМЕНТОВ.....	8
ЗАКЛЮЧЕНИЕ	26
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	27

ВВЕДЕНИЕ

Хранение, обработка и анализ больших объемов данных стало неотъемлемой частью каждой компании и организации, которые хотят получить ценные знания о рынке и преимущество перед конкурентами.

В рамках выполнения данной курсовой работы был реализован конвейер обработки данных, использующий различные технологии и инструменты работы с большими данными, включая HDFS, Sqoop, MariaDB, Hive, Spark. Полученный конвейер позволяет эффективно хранить и обрабатывать большие объемы данных.

Тема данной курсовой работы актуальна для крупных компаний и организаций в любой сфере бизнеса.

Цель курсовой работы — построить конвейер обработки данных.

Задачи, решаемые в данной курсовой работе:

- взаимодействие с VK API для получения исходных данных;
- построение конвейера обработки данных;
- проведен качественный анализ данных о пользователях;
- отработано качественное оформление документации.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Описание используемых инструментов

HDFS (Hadoop Distributed File System) — это распределенная файловая система для хранения и обработки больших объемов данных на кластерах серверов. Она обеспечивает высокую пропускную способность и отказоустойчивость, состоя из NameNode и DataNode.

Sqoop — инструмент для передачи данных между системами баз данных и Hadoop, позволяя эффективно импортировать и экспортировать данные из баз данных в Hadoop и обратно.

PySpark — Python API для Apache Spark, предназначенная для обработки больших объемов данных. Она позволяет использовать мощные возможности Spark, такие как распределенные вычисления, обработка потоковых данных, машинное обучение и графовые алгоритмы, с использованием языка программирования Python.

YARN (Yet Another Resource Negotiator) — основной компонент Apache Hadoop, представляющий собой среду управления ресурсами и планирования задач для выполнения вычислений в распределенной среде.

MariaDB — открытая реляционная база данных, которая обеспечивает надежное хранение и управление структурированными данными.

Hive — инфраструктура для обработки больших данных, построенная поверх Hadoop. Она предоставляет SQL-подобный язык запросов, называемый HiveQL, для выполнения аналитических запросов и обработки данных в Hadoop.

Эти инструменты обеспечивают мощные возможности для хранения, обработки и анализа больших объемов данных в распределенных средах, играя важную роль в различных сценариях обработки данных и аналитики.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

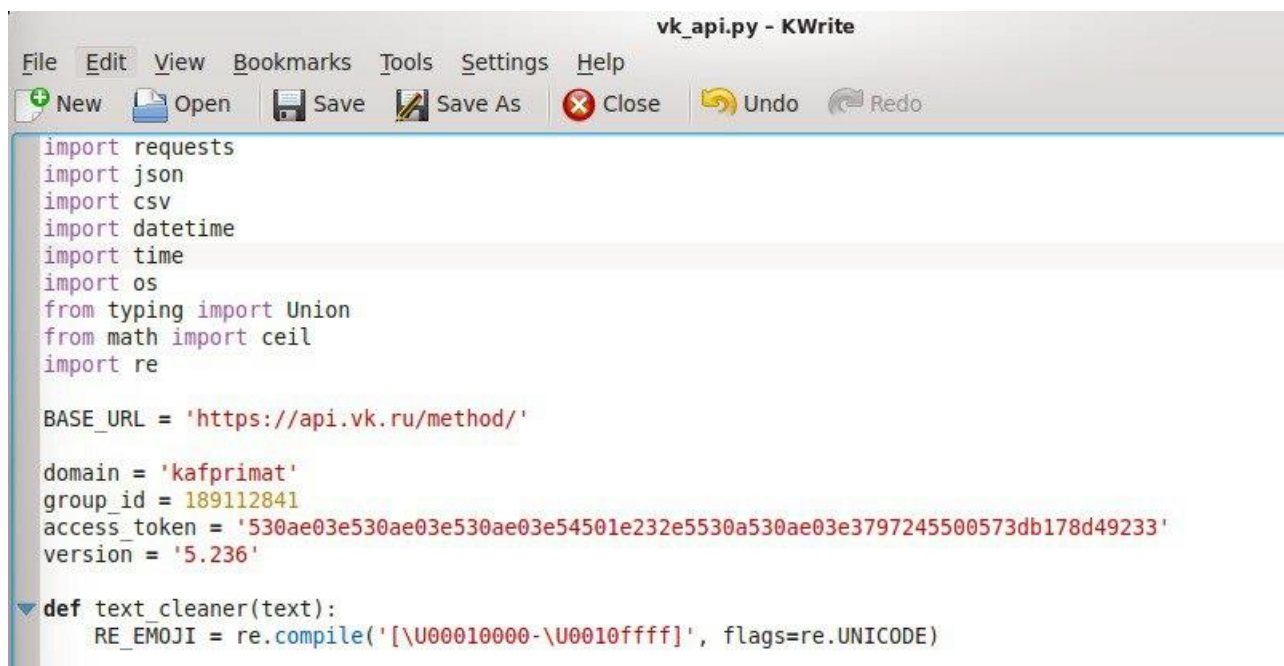
2.1 Установка, конфигурация и применение инструментов

Курсовая работа будет написана с использованием программы для виртуализации «Oracle VM Virtualbox». В первую очередь необходимо получить данные для работы. Для этого создадим директорию для курсовой работы, в которой создадим файл в формате .py под скрипт для сбора данных.

```
[student@desktop-mr5fiet ~]$ mkdir cursovaya
[student@desktop-mr5fiet ~]$ cd cursovaya
[student@desktop-mr5fiet cursovaya]$ touch vk_api.py
[student@desktop-mr5fiet cursovaya]$ ls
vk_api.py
[student@desktop-mr5fiet cursovaya]$
```

Рисунок 2.1 — Создание директории

В редакторе KWrite напишем python-скрипт для работы с VK API и сбора информации о пользователях и постах из группы.



The screenshot shows the KWrite text editor with the title bar 'vk_api.py - KWrite'. The menu bar includes File, Edit, View, Bookmarks, Tools, Settings, and Help. The toolbar contains icons for New, Open, Save, Save As, Close, Undo, and Redo. The script content is as follows:

```
import requests
import json
import csv
import datetime
import time
import os
from typing import Union
from math import ceil
import re

BASE_URL = 'https://api.vk.ru/method/'

domain = 'kafprimat'
group_id = 189112841
access_token = '530ae03e530ae03e530ae03e54501e232e5530a530ae03e3797245500573db178d49233'
version = '5.236'

def text_cleaner(text):
    RE_EMOJI = re.compile('[\U00010000-\U0010ffff]', flags=re.UNICODE)
```

Рисунок 2.2 — Python-скрипт для сбора данных

Первым шагом стало получение списка всех пользователей группы и их уникальных идентификаторов. Для этого была реализована функция, которая извлекает данные всех пользователей группы с помощью VK API.

```
def get_all_users(): # функция получения id всех пользователей группы (можно еще получить какие-то дан
    all_user_ids = []

    offset = 0
    while offset < 2000:
        users = requests.get(url=f'{BASE_URL}groups.getMembers', params={
            'group_id': domain,
            'sort': 'id_asc',
            'count': '1000',
            'offset': offset,
            'access_token': access_token,
            'v': version
        })
        user_ids = users.json()['response']['items'] # все подписчики группы
        offset += 1000
        all_user_ids.extend(user_ids)

    all_user_ids = [str(id) for id in all_user_ids] # id всех пользователей группы
    return all_user_ids
```

Рисунок 2.3 — Получение списка пользователей

Затем необходимо получить данные о дате рождения, возрасте, городе, поле, имени, фамилии и других параметрах каждого пользователя группы, используя соответствующий метод VK API. Для реализации данного функционала была разработана функция "get_users_info", которая принимает на вход список уникальных идентификаторов пользователей группы и на выходе записывает всю информацию о пользователях в формате CSV, а также формирует список активных пользователей группы.

```
def get_users_info(user_ids: list) -> list: # функция получения информации о списке пользовател
    user_info = []
    count = 500
    left = 0
    right = count + 1
    for i in range(ceil(len(user_ids) / count)):
        time.sleep(0.5)
        data = requests.get(url=f'{BASE_URL}users.get', params={
            'user_ids': ','.join(user_ids[left:right]),
            'fields': 'sex,bdate,city,country,verified,last_seen,personal',
            'access_token': access_token,
            'v': version,
            'lang': 0
        })
        response = data.json()['response']

        result = [[info['id'],
            user_info.extend(result)

        left += count
        right += count

    # запись в файл
    with open('users.csv', 'w', encoding='utf-8') as file:
        writer = csv.writer(file)
        writer.writerow(['ID', 'bdate', 'age', 'city', 'country', 'last_seen_platform', 'last_s
        writer.writerows(user_info)

    return user_info
```

Рисунок 2.4 — Получение данных о пользователях

Далее необходимо получить информацию о всех постах пользователей на личных страницах с 2019 года. Все полученные данные (дата поста, кол-во лайков, комментариев, репостов, просмотров и фотографий) записываются в формате CSV. Однако многие пользователи закрывают свою страницу, из-за чего информацию о постах у таких пользователей получить невозможно.

```
def get_user_posts(user_id: int) -> list: # Получаем все посты с страницы пользователей
    print(f'get_user_posts {user_id}')
    post_list = []
    offset = 0
    count = 100
    while offset < count:
        time.sleep(0.4)
        data = requests.get(url=f"{BASE_URL}wall.get", params={
            'owner_id': user_id,
            'count': 100,
            'offset': offset,
            'filter': 'owner',
            'access_token': access_token,
            'v': version
        })

        # private wall handler
        if 'error' in data.json().keys():
            return data.json()

        response = data.json()['response']

        count = response['count']
        items = response['items']

        result = [[user_id,
                    post_list.extend(result)

        offset += 100
```

Рисунок 2.4 — Получение постов пользователей

Затем необходимо получить список всех постов в группе. Функция "get_group_posts" получает такую информацию, как дату поста, количество лайков, комментариев, репостов, просмотров записи, количество фотографий и текст поста. Все данные записываются в формате CSV.

```
def get_group_posts(): # получаем список постов группы
    post_list = []
    ids = []
    offset = 0
    count = 1000
    while offset < count:
        time.sleep(0.5)
        data = requests.get(url=f"{BASE_URL}wall.get", params={
            'domain': 'kafrimat',
            'count': 100,
            'offset': offset,
            'extended': 1,
            'filter': 'all',
            'access_token': access_token,
            'v': version
        })

        response = data.json()['response']

        count = response['count']
        items = response['items']

        ids_ = [item['id'] for item in items]

        result = [[item['id'],
                    post_list.extend(result)
                    ids.extend(ids_)
                    offset += 100
```

Рисунок 2.5 — Получение списка постов в группе

Заключительным этапом сбором данных является получение списка пользователей, поставивших лайк на каждую запись в группе. Итоговые данные записываются в формате CSV.

```
def get_post_likes(post_id: int):# получаем отношение id постов и id пользователей, лайкнувших пост
    print(f'Обработка поста id: {post_id}')
    time.sleep(0.4)

    data_likes = requests.get(url=f"{BASE_URL}likes.getList", params={
        'type': 'post',
        'owner_id': f'--{group_id}',
        'item_id': post_id,
        'count': 1000,
        'access_token': access_token,
        'v': version
    })

    try:
        likes_ids = data_likes.json()['response']['items']
    except:
        print(data_likes.json())

    rows = zip([post_id for i in range(len(likes_ids))], likes_ids)

    with open('post_likes.csv', 'a') as file:

    return None
```

Рисунок 2.6 — Лайки пользователей в группе

Начнем выполнение скрипта. Все данные автоматически будут собираться в файлы в выбранной директории.

```
[student@desktop-mr5fiet cursovaya]$ python3 vk_api.py
Обработка поста id: 2980
Обработка поста id: 3066
Обработка поста id: 3064
Обработка поста id: 3062
```

Рисунок 2.7 — Сбор данных

Все собранные данные хранятся в локальной файловой системе в формате CSV.



Рисунок 2.8 — Исходные данные

Импортируем данные из локальной файловой системы в HDFS. Затем проверим наличие исходных данных в HDFS.

```
[student@desktop-mr5fiet ~]$ hdfs dfs -put ~/cursovaya/users.csv
[student@desktop-mr5fiet ~]$ hdfs dfs -put ~/cursovaya/user_posts.csv
[student@desktop-mr5fiet ~]$ hdfs dfs -put ~/cursovaya/posts.csv
[student@desktop-mr5fiet ~]$ hdfs dfs -put ~/cursovaya/post_likes.csv
[student@desktop-mr5fiet ~]$ hdfs dfs -ls
Found 6 items
drwxr-xr-x  - student student      0 2021-08-28 23:21 .hiveJars
-rw-r--r--  1 student student    4987 2021-08-19 22:27 data.txt
-rw-r--r--  1 student student  168446 2024-06-04 22:35 post_likes.csv
-rw-r--r--  1 student student  789390 2024-06-04 22:35 posts.csv
-rw-r--r--  1 student student 1136884 2024-06-04 22:34 user_posts.csv
-rw-r--r--  1 student student  120339 2024-06-04 22:34 users.csv
[student@desktop-mr5fiet ~]$
```

Рисунок 2.1 — Импорт исходных данных в HDFS

Перед импортированием данных из HDFS в MariaDB необходимо авторизоваться в РСУБД и создать новую базу данных "vk_api", в которой создадим таблицы для каждого CSV файла.

```
[student@desktop-mr5fiet ~]$ mysql --user=student --password=student
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 85
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database vk_api;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| hive |
| hue |
| labs |
| mysql |
| performance_schema |
| vk_api |
+-----+
7 rows in set (0.00 sec)
```

Рисунок 2.10 — Создание базы данных в MariaDB

Создадим таблицу для импорта данных о постах в группе из файла posts.csv.

```
MariaDB [vkApi]> create table posts (  
-> post_id int primary key,  
-> date datetime,  
-> likes int,  
-> comments int,  
-> reposts int,  
-> views int,  
-> attachments int,  
-> text_len int  
-> );
```

Рисунок 2.11 — Таблица posts

Начинаем экспорт в таблицу posts в MariaDB.

```
[student@localhost cursovaya]$ sqoop export --connect jdbc:mysql://localhost/vkApi --username student --password student --table posts --export-dir posts.csv --fields-terminated-by ',' --lines-terminated-by '\n' --input-null-string '' --input-null-non-string ''
```

Рисунок 2.12 — Экспорт данных

Пример выборки данных из таблицы posts.

```
MariaDB [vkApi]> select * from posts;
```

post_id	date	likes	comments	reposts	views	attachments	text_len
10	2019-12-03 17:40:42	3	0	0	230	0	134
11	2019-12-03 17:47:37	3	0	0	407	0	152
15	2019-12-26 17:24:47	17	1	0	409	1	25
26	2020-03-06 23:57:47	2	0	0	188	1	58
39	2020-03-28 02:04:23	2	0	0	271	0	1
40	2020-04-11 03:33:26	2	0	0	227	0	1
41	2020-04-14 22:09:02	2	0	0	240	0	51
44	2020-04-21 03:39:22	4	0	1	303	1	77
45	2020-04-22 21:16:42	2	0	0	253	3	148
50	2020-04-30 01:45:48	3	0	0	324	0	96

Рисунок 2.13 — Пример выборки данных

Создадим таблицу пользователей группы и информации о них. Экспортируем данные из файла users.csv в таблицу users в MariaDB. Выведем часть данных на экран для проверки.

```

MariaDB [vkApi]> create table users (
-> user_id int primary key,
-> bdate date,
-> age int,
-> city varchar(30),
-> country varchar(30),
-> last_seen_platform varchar(30),
-> last_seen_time datetime,
-> sex char,
-> verified int,
-> first_name varchar(30),
-> last_name varchar(30),
-> is_closed int
-> );

```

Рисунок 2.14 — Таблица users

```

[student@localhost cursovaya]$ sqoop export --connect jdbc:mysql://localhost/vkApi --username student --password student --table users --export-dir users.csv --fields-terminated-by ',' --lines-terminated-by '\n' --input-null-string '' --input-null-non-string '' --update-key 'user_id' --update-mode 'allowinsert'
Warning: /usr/local/sqoop/sqoop-1.4.7/./hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/sqoop-1.4.7/./accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.

```

Рисунок 2.15 — Экспорт данных

```

MariaDB [vkApi]> select * from users limit 10;

```

user_id	bdate	age	city	country	last_seen_platform	last_seen_time	sex	verified	first_name	last_name	is_closed
303206	2000-09-16	NULL	Moscow	Russia	Android app	2024-06-07 13:30:54	M	0	Andrey	Rinchino	1
989742	2000-04-08	34	Moscow	Russia	Full version	2024-06-07 17:01:31	F	0	Irina	Lebed	0
4510163	2000-11-18	NULL	Moscow	Russia	iPhone app	2024-06-07 17:02:48	F	0	Printsessa	Olga	0
5763938	2000-06-22	19	Moscow	Russia	Android app	2024-06-07 17:13:15	F	0	Anna	Solovyova	0
8100016	2000-09-21	30	Moscow	Russia	Full version	2024-06-06 21:32:50	F	0	Diana	Sarkisyan	0
10834741	2000-06-12	NULL	Dzerzhinsky	Russia	iPhone app	2024-06-07 17:07:06	M	0	Anatoly	Sinitsyn	0
15859585	2000-10-04	NULL	Moscow	Russia	Android app	2024-06-07 17:04:41	F	0	Sashka	Mayakovskaya	1
18037347	2000-05-08	NULL	Moscow	Russia	iPhone app	2024-06-07 17:07:59	F	0	Anna	Paukova	0
18459623	2000-05-16	NULL	Lyubertsy	Russia	iPhone app	2024-06-07 14:46:35	F	0	Ksyusha	Lebedeva	1
20732517	2000-03-16	NULL	NULL	Russia	Full version	2024-06-07 17:13:42	F	0	Elina	Lavrinenko	0

```

10 rows in set (0.01 sec)

```

Рисунок 2.16 — Пример выборки данных

Создадим таблицу с пользователями и их лайками к постам. Начнем экспорт и выведем часть данных на экран.

```

MariaDB [vkApi]> create table post_likes (
-> post_id int,
-> user_id int
-> );

```

Рисунок 2.17 — Таблица post_likes

```

[student@localhost cursovaya]$ sqoop export --connect jdbc:mysql://localhost/vkApi --username student --password student --table post_likes --export-dir post_likes.csv --fields-terminated-by ',' --lines-terminated-by '\n' --input-null-string '' --input-null-non-string ''
Warning: /usr/local/sqoop/sqoop-1.4.7/./hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/sqoop-1.4.7/./accumulo does not exist! Accumulo imports will fail.

```

Рисунок 2.18 — Экспорт данных


```

MariaDB [vkApi]> select * from post_likes where post_id = 2980;
+-----+-----+
| post_id | user_id |
+-----+-----+
| 2980    | 260875449 |
| 2980    | 6052217  |
| 2980    | 85717035 |
| 2980    | 183809496 |
| 2980    | 252982058 |
| 2980    | 280831516 |
| 2980    | 393077771 |
| 2980    | 486032871 |
| 2980    | 559498191 |
| 2980    | 741760146 |
+-----+-----+
10 rows in set (0.00 sec)

```

Рисунок 2.19 — Пример выборки данных

Создадим таблицу с информацией о постах пользователей на личных страницах.

```

MariaDB [vkApi]> create table user_posts (
-> user_id int,
-> post_id int,
-> post_date datetime,
-> likes int,
-> comments int,
-> reposts int,
-> views int,
-> attachments int,
-> text_len int
-> );

```

Рисунок 2.20 — Таблица post_likes

```

[student@localhost cursovaya]$ sqoop export --connect jdbc:mysql://localhost/vkApi --username student --password student --table user_posts --export-dir user_posts.csv --fields-terminated-by ',' --lines-terminated-by '\n' --input-null-string '' --input-null-non-string ''
Warning: /usr/local/sqoop/sqoop-1.4.7/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.

```

Рисунок 2.21 — Экспорт данных

```

MariaDB [vkApi]> select * from user_posts where user_id = 8100016;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | post_id | post_date          | likes | comments | reposts | views | attachments | text_len |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 8100016 | 12169   | 2022-05-27 03:04:39 | 47    | 0        | 0       | 1676  | 1           | 33        |
| 8100016 | 12192   | 2024-06-01 15:33:34 | 0     | 0        | 0       | 139   | 0           | 3          |
| 8100016 | 12191   | 2023-10-13 03:26:26 | 93    | 0        | 1       | 482   | 1           | 1          |
| 8100016 | 12190   | 2023-09-22 04:41:48 | 73    | 0        | 0       | 761   | 3           | 39         |
| 8100016 | 12189   | 2023-09-18 05:51:10 | 84    | 0        | 0       | 884   | 1           | 6          |
| 8100016 | 12188   | 2023-06-17 04:09:50 | 87    | 0        | 4       | 2387  | 1           | 37         |
| 8100016 | 12185   | 2022-09-23 15:36:19 | 68    | 0        | 2       | 639   | 1           | 1          |
| 8100016 | 12183   | 2022-09-22 05:41:17 | 53    | 0        | 1       | 678   | 2           | 93         |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

```

Рисунок 2.22 — Пример выборки данных

Затем необходимо перенести данные из MariaDB в Hive, используя фреймворк sqoop. Для начала сделаем базу данных в Hive.

```
0: jdbc:hive2://> create database data_vkApi;
24/06/07 20:27:32 [HiveServer2-Background-Pool: Three] OK
Database hive.data_vkApi, returning NoSuchObjectException
OK
No rows affected (0.062 seconds)
0: jdbc:hive2://>
0: jdbc:hive2://>
0: jdbc:hive2://> show databases;
OK
+-----+
| database_name |
+-----+
| data_vkapi    |
+-----+
```

Рисунок 2.23 — Создание базы данных в Hive

Перенос таблицы users из MariaDB в Hive.

```
[student@localhost cursovaya]$ sqoop import --connect jdbc:mysql://localhost/vkApi --username student --password student --table users --hive-import --hive-database data_vkapi --hive-table users
Warning: /usr/local/sqoop/sqoop-1.4.7/./hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/sqoop-1.4.7/./accumulo does not exist! Accumulo imports will fail.
```

Рисунок 2.24 — Перенос таблицы users

Перенос таблицы posts из MariaDB в Hive.

```
[student@localhost cursovaya]$ sqoop import --connect jdbc:mysql://localhost/vkApi --username student --password student --table posts --hive-import --hive-database data_vkapi --hive-table posts
Warning: /usr/local/sqoop/sqoop-1.4.7/./hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/sqoop-1.4.7/./accumulo does not exist! Accumulo imports will fail.
```

Рисунок 2.25 — Перенос таблицы posts

Перенос таблицы user_posts из MariaDB в Hive.

```
[student@localhost cursovaya]$ sqoop import --connect jdbc:mysql://localhost/vkApi --username student --password student --table user_posts --hive-import --hive-database data_vkapi --hive-table user_posts --split-by user_id
Warning: /usr/local/sqoop/sqoop-1.4.7/./hcatalog does not exist! HCatalog jobs will fail.
```

Рисунок 2.26 — Перенос таблицы user_posts

Перенос таблицы post_likes из MariaDB в Hive.

```
[student@localhost cursovaya]$ sqoop import --connect jdbc:mysql://localhost/vkApi --username student --password student --table post_likes --hive-import --hive-database data_vkapi --hive-table post_likes --split-by post_id
Warning: /usr/local/sqoop/sqoop-1.4.7/./hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop/sqoop-1.4.7/./accumulo does not exist! Accumulo imports will fail.
```

Рисунок 2.27 — Перенос таблицы post_likes

Проверим, что все таблицы были успешно перенесены из MariaDB в Hive.

```
0: jdbc:hive2://> show tables;
OK
+-----+
| tab_name |
+-----+
| post_likes |
| posts |
| user_posts |
| users |
+-----+
4 rows selected (0.04 seconds)
```

Рисунок 2.28 — Вывод таблиц

Запустим Pyspark для анализа данных.

```
[student@localhost cursovaya]$ pyspark
[I 00:34:48.836 NotebookApp] Serving notebooks from local directory: /home/student/cursovaya
[I 00:34:48.836 NotebookApp] Jupyter Notebook 6.4.3 is running at:
[I 00:34:48.836 NotebookApp] http://localhost:3333/?token=4f59477bbadd52e2534475e20f0928002d9b317da9cdab2d
[I 00:34:48.836 NotebookApp] or http://127.0.0.1:3333/?token=4f59477bbadd52e2534475e20f0928002d9b317da9cdab2d
[I 00:34:48.836 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 00:34:48.954 NotebookApp]
```

Рисунок 2.29 — Запуск Pyspark

Проверим список таблиц.

```
In [4]: tables = spark.sql("show tables")
        tables.show()

+-----+-----+-----+
| database | tableName | isTemporary |
+-----+-----+-----+
| data_vkapi | post_likes | false |
| data_vkapi | posts | false |
| data_vkapi | user_posts | false |
| data_vkapi | users | false |
+-----+-----+-----+
```

Рисунок 2.31 — Список таблиц

Прочитаем все таблицы из Hive и выведем несколько значений для проверки правильности переноса данных.

```
55]: usersDF = spark.read.table("data_vkapi.users")
      usersDF.show(5)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | bdate | age | city | country | last_seen_platform | last_seen_time | sex | verified | first_name | last_name | i |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 303206 | 2000-09-16 | null | Moscow | Russia | Android app | 2024-06-07 13:30:... | M | 0 | Andrey | Rinchino | 1 |
| 989742 | 2000-04-08 | 34 | Moscow | Russia | Full version | 2024-06-07 17:01:... | F | 0 | Irina | Lebed | 0 |
| 4510163 | 2000-11-18 | null | Moscow | Russia | iPhone app | 2024-06-07 17:02:... | F | 0 | Printsessa | Olga | 0 |
| 5763938 | 2000-06-22 | 19 | Moscow | Russia | Android app | 2024-06-07 17:13:... | F | 0 | Anna | Solovyova | 0 |
| 8100016 | 2000-09-21 | 30 | Moscow | Russia | Full version | 2024-06-06 21:32:... | F | 0 | Diana | Sarkisyan | 0 |
```

Рисунок 2.32 — Таблица users

```
In [66]: postsDF = spark.read.table("data_vkapi.posts")
postsDF.show(5)
```

post_id	date	likes	comments	reposts	views	attachments	text_len
10	2019-12-03 17:40:...	3	0	0	230	0	134
11	2019-12-03 17:47:...	3	0	0	407	0	152
15	2019-12-26 17:24:...	17	1	0	409	1	25
26	2020-03-06 23:57:...	2	0	0	188	1	58
39	2020-03-28 02:04:...	2	0	0	271	0	1

Рисунок 2.33 — Таблица posts

```
In [67]: post_likesDF = spark.read.table("data_vkapi.post_likes")
post_likesDF.show(5)
```

post_id	user_id
764	56715320
764	188563975
764	229991524
764	269074055
758	56715320

only showing top 5 rows

Рисунок 2.34 — Таблица post_likes

```
In [68]: user_postsDF = spark.read.table("data_vkapi.user_posts")
user_postsDF.show(5)
```

user_id	post_id	post_date	likes	comments	reposts	views	attachments	text_len
8100016	12169	2022-05-27 03:04:...	47	0	0	1676	1	33
8100016	12192	2024-06-01 15:33:...	0	0	0	139	0	3
8100016	12191	2023-10-13 03:26:...	93	0	1	482	1	1
8100016	12190	2023-09-22 04:41:...	73	0	0	761	3	39
8100016	12189	2023-09-18 05:51:...	84	0	0	884	1	6

only showing top 5 rows

Рисунок 2.35 — Таблица user_posts

Выведем 10 самых активных пользователей. Активность определяется подсчетом количества постов на личной странице пользователя с 2019 года.

```
# find 10 most active users. Activity is defined by the amount of posts on the page
users = usersDF.toPandas()
user_posts = user_postsDF.toPandas()

posts_amount = user_posts.groupby('user_id').size().reset_index().rename(columns={0: 'post_number'})

users_info = users.merge(posts_amount, on='user_id', how='left')
top10 = users_info.sort_values('post_number', ascending=False).head(10)
top10['FI'] = top10[['first_name', 'last_name']].apply(lambda x: ' '.join(x), axis=1)

ax = top10.plot.barh(x='FI', y='post_number', figsize=(12, 10))
ax.set_title("10 most active users", fontsize=20)
ax.tick_params(labelsize=15)
ax.set_xlabel('Number of posts on the page', fontsize=15)
ax.set_ylabel('', fontsize=15)
```

Рисунок 2.36 — Код для получения самых активных пользователей

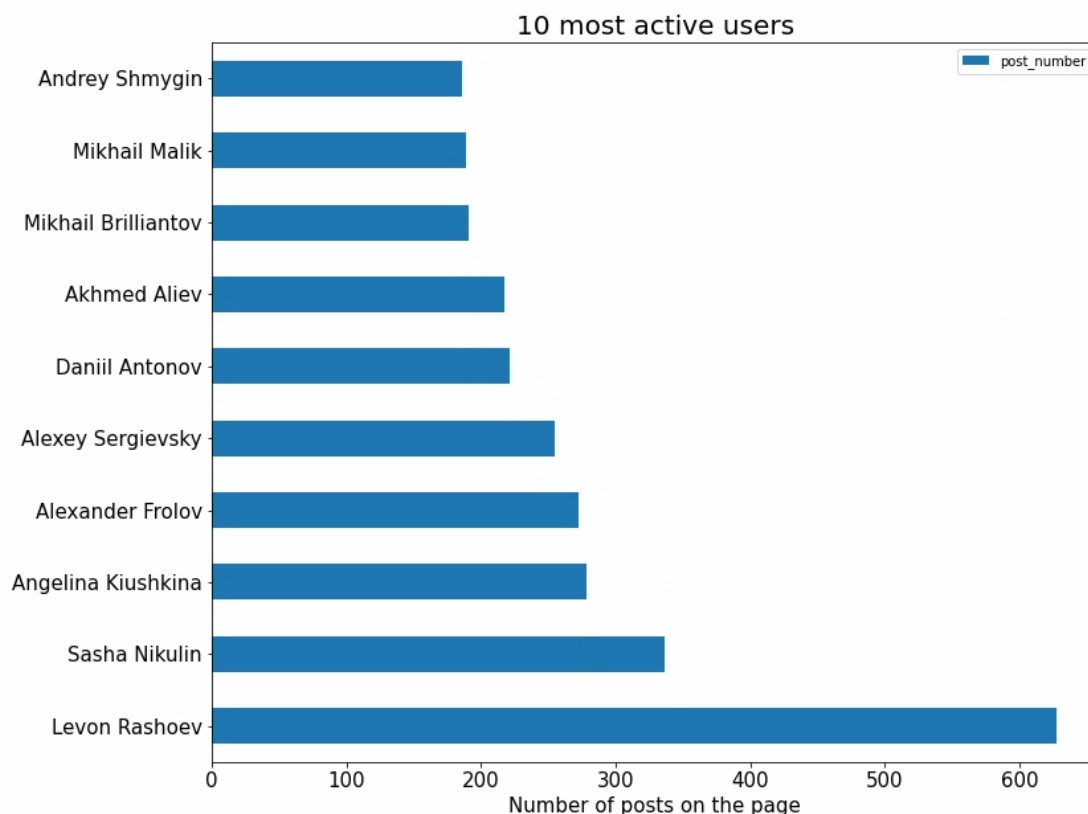


Рисунок 2.37 — Столбчатая диаграмма 10 самых активных пользователей

Теперь необходимо выявить степень корреляции между активностью пользователя на личной странице и в группе. За активность пользователя в группе будем считать количество поставленным им лайков на записи в группе.

По результатам можно определить отсутствие выраженной корреляции.

```
# correlation between activity on page and amount of likes in group
post_likes = post_likesDF.toPandas()

likes = post_likes.groupby('user_id').size().reset_index().rename(columns={0:'likes_number'})
likes
users_info_all = users_info.merge(likes, on='user_id', how='left')

info_filtered = users_info_all[users_info_all[['post_number', 'likes_number']].notnull().all(1)]

import seaborn as sns

plt.figure(figsize=(8, 8))
ax = sns.heatmap(info_filtered.iloc[:, -2:].corr(), annot=True, cmap='Blues')
ax.set_title("Correlation between activity in group and on personal page", fontsize=15)
ax.tick_params(labelsize=12)
plt.show()
```

Рисунок 2.38 — Код корреляции между активностью в группе и на личной странице

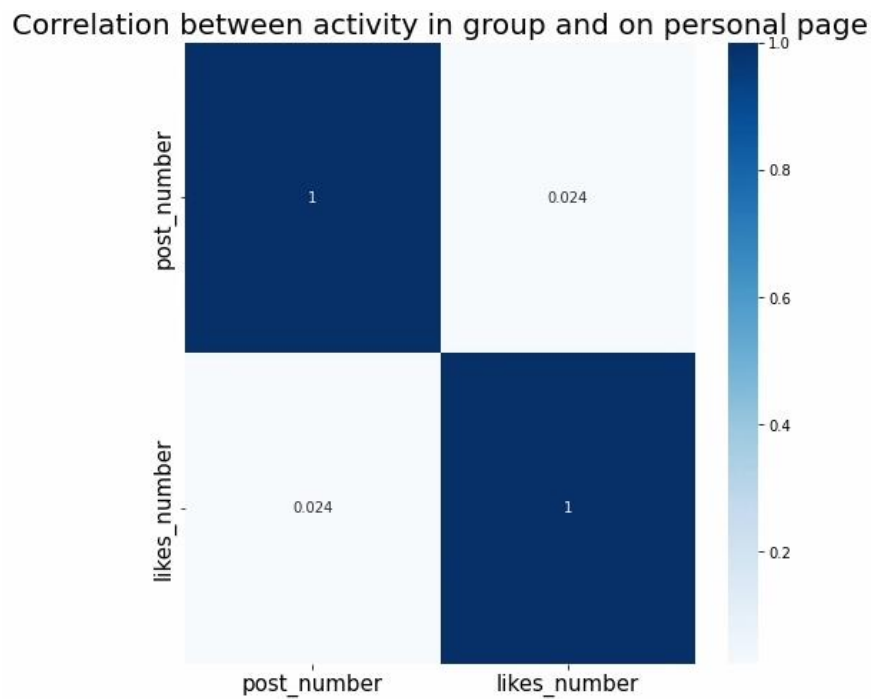


Рисунок 2.39 — Тепловая карта корреляции между активностью в группе и на личной странице

Построим матрицу корреляции между длиной текста от количества лайков, комментариев, репостов, просмотров и прикрепленных фотографий к записи.

```
# correlation between length length of posts and likes, comments and views
posts = postsDF.toPandas()

posts_data = posts.loc[:, ~posts.columns.isin(['date'])].set_index('post_id')
corr_df = posts_data.corr()

plt.figure(figsize=(10, 8))
ax = sns.heatmap(corr_df, annot=True, cmap='Blues')
ax.tick_params(labelsize=12)
ax.set_title("Correlation matrix of posts features", fontsize=15)
plt.show()
```

Рисунок 2.40 — Код для построения матрицы корреляций

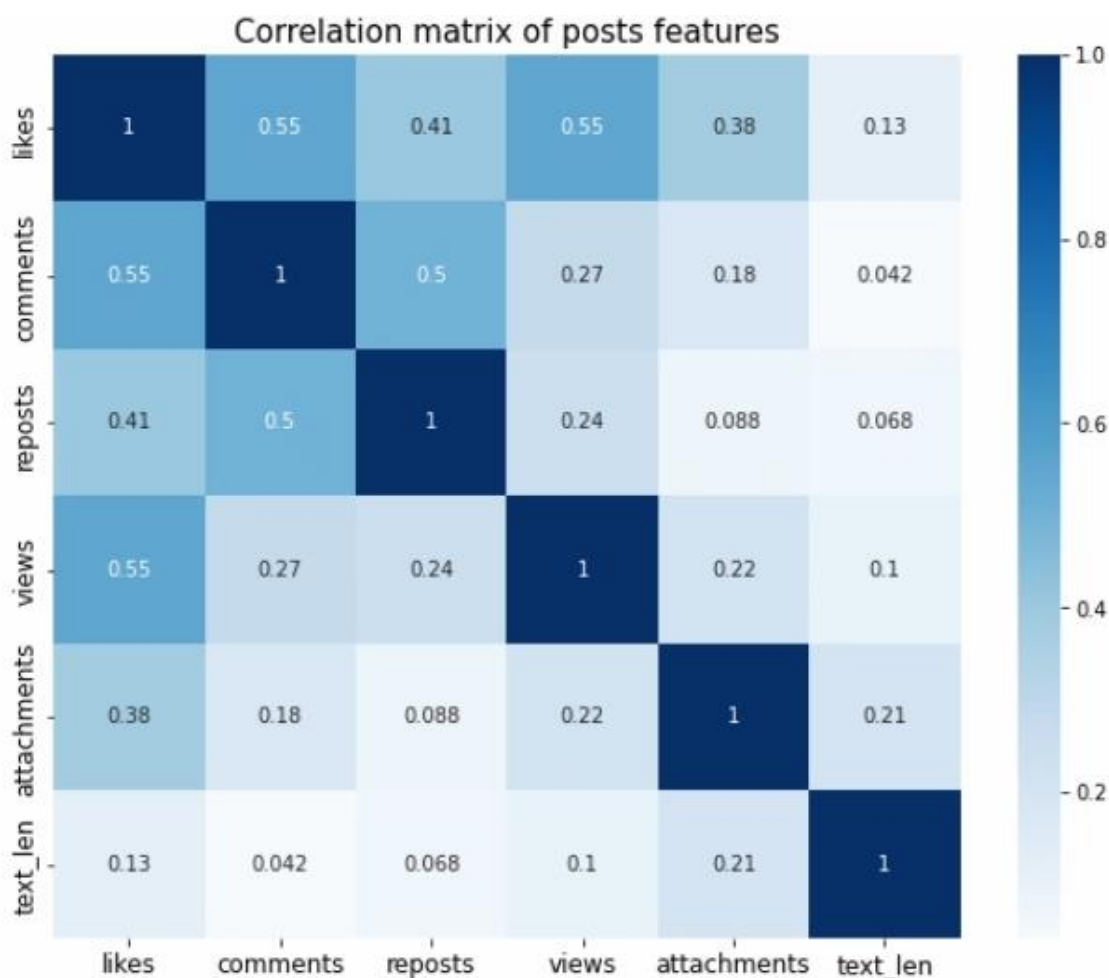


Рисунок 2.41 — Матрица корреляций

По результатам можно отметить отсутствие значимой корреляции между длиной записи и другими характеристиками записи. Однако имеется слабая положительная корреляция между количеством лайков и комментариев с просмотрами, а также между количеством комментариев и репостов записи.

Теперь необходимо кластеризовать записи в группе по лайкам, комментариям, просмотрам, репостам и количеству фотографий у записи.

Для начала необходимо произвести нормализацию данных, используя метод `minmax`-нормализации. Затем выполним кластеризацию данных методом DBSCAN и посмотрим сколько записей относится к каждому кластеру.


```

from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import DBSCAN
from sklearn.manifold import TSNE
import numpy as np

scaler = MinMaxScaler()
norm_data = pd.DataFrame(scaler.fit_transform(posts_data), columns= posts_data.columns) # нормализованные данные

DBScan = DBSCAN(eps=0.1, min_samples=20, metric='l1')
DB = DBScan.fit_predict(norm_data)

unique, counts = np.unique(DB, return_counts=True)

fig, ax = plt.subplots(figsize=(10, 10))
ax.pie(counts, labels=['-1 cluster', '0 cluster', '1 cluster'], autopct='%1.1f%%', textprops={'fontsize':14})
ax.set_title("Amount of posts in each cluster", fontsize=20)
plt.show()

```

Рисунок 2.42 — Код кластеризации

После выполнения алгоритма получили 3 кластера. Выведем количество постов в каждом кластере.

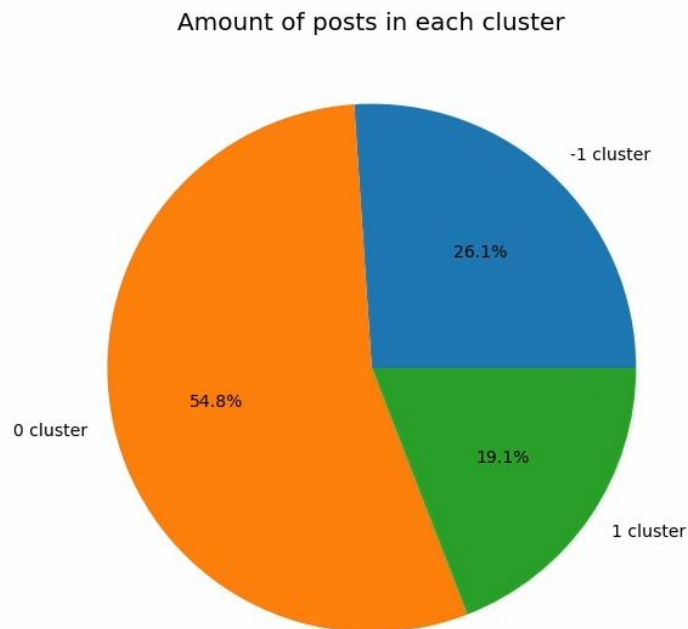


Рисунок 2.43 — Количество постов в кластерах

Затем необходимо визуализировать полученные результаты. Для этого произведем снижение размерности исходных нормализованных данных методом t-sne.

```

: # T-SNE visualisation
X_embedded = TSNE(n_components=2, learning_rate='auto', perplexity=130, random_state=101).fit_transform(norm_data)

fig, ax = plt.subplots(figsize=(10, 10))
ax.scatter(X_embedded[:, 0], X_embedded[:, 1], s=60, c=DB, marker='o', cmap = 'Paired', edgecolors='black')
ax.set_title("T-SNE DBSCAN", fontsize=20)
ax.tick_params(labelsize=15)
plt.show()

```

Рисунок 2.44 — Код снижения размерности и визуализации

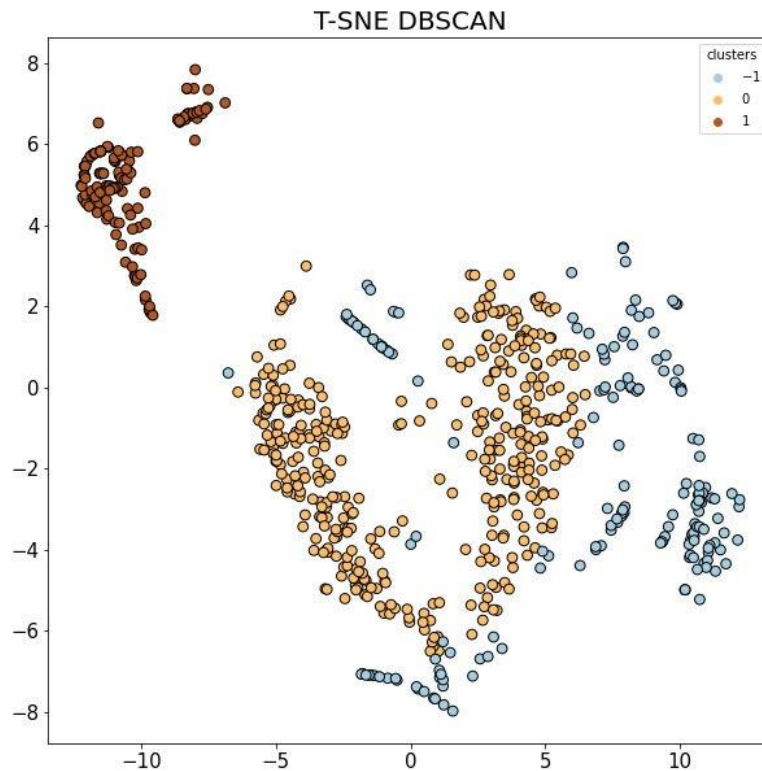


Рисунок 2.45 — График рассеяния полученных кластеров

Далее необходимо исследовать посты внутри каждого кластера. Для этого выведем и сравним диаграммы размаха для каждого кластера по характеристикам постов.

```
posts_data['cluster'] = DB

plt.figure(figsize=(15,8))
ax = sns.boxplot(data=posts_data, x='cluster', y='likes', hue='cluster')
ax.tick_params(labelsize=15)
ax.set_xlabel('cluster', fontsize=15)
ax.set_ylabel('likes', fontsize=15)
ax.set_ylim([0, 80])
ax.set_title("Likes distribution in cluster", fontsize=20)
```

Рисунок 2.46 — Код построения диаграммы размаха лайков по кластерам

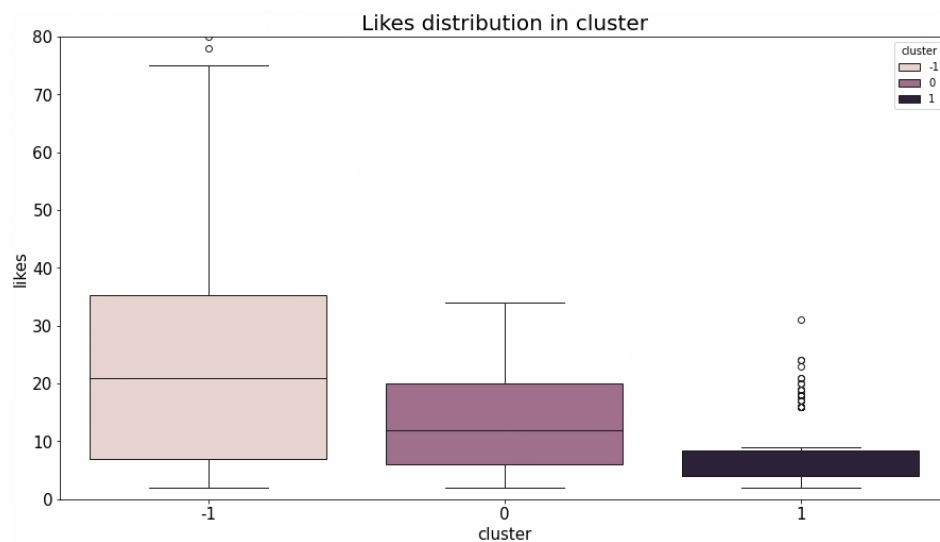


Рисунок 2.47 — Диаграмма размаха лайков по кластерам

```
plt.figure(figsize=(15,8))
ax = sns.boxplot(data=posts_data, x='cluster', y='reposts', hue='cluster')
ax.tick_params(labelsize=15)
ax.set_xlabel('cluster', fontsize=15)
ax.set_ylabel('reposts', fontsize=15)
ax.set_title("Reposts distribution in cluster", fontsize=20)
ax.set_ylim([0, 20])
```

Рисунок 2.48 — Код построения диаграммы размаха репостов по кластерам

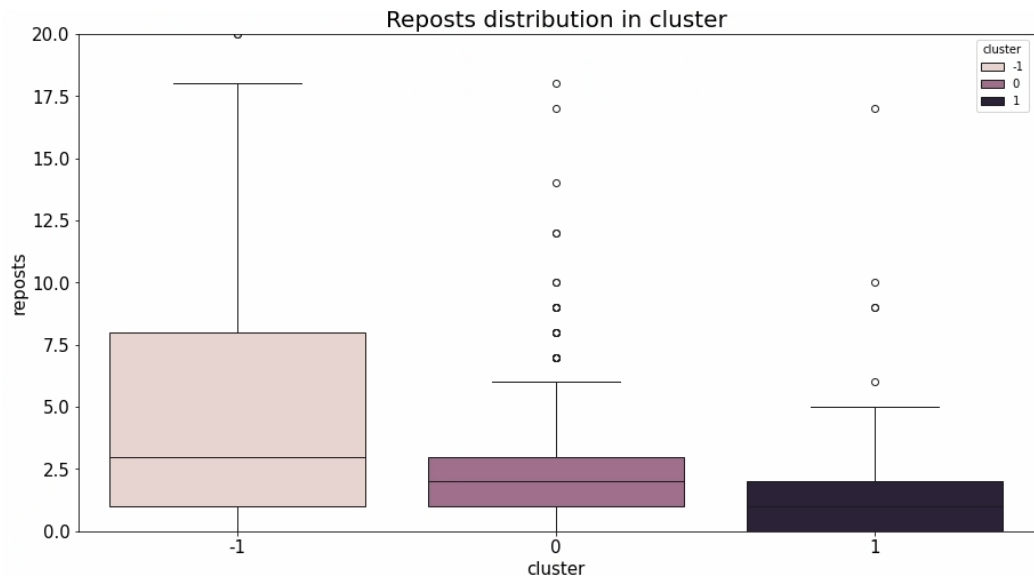


Рисунок 2.49 — Диаграмма размаха репостов по кластерам

```
plt.figure(figsize=(15,8))
ax = sns.boxplot(data=posts_data, x='cluster', y='views', hue='cluster')
ax.tick_params(labelsize=15)
ax.set_xlabel('cluster', fontsize=15)
ax.set_ylabel('views', fontsize=15)
ax.set_title("Views distribution in cluster", fontsize=20)
ax.set_ylim([0, 2500])
```

Рисунок 2.50 — Код построения диаграммы размаха просмотров по кластерам

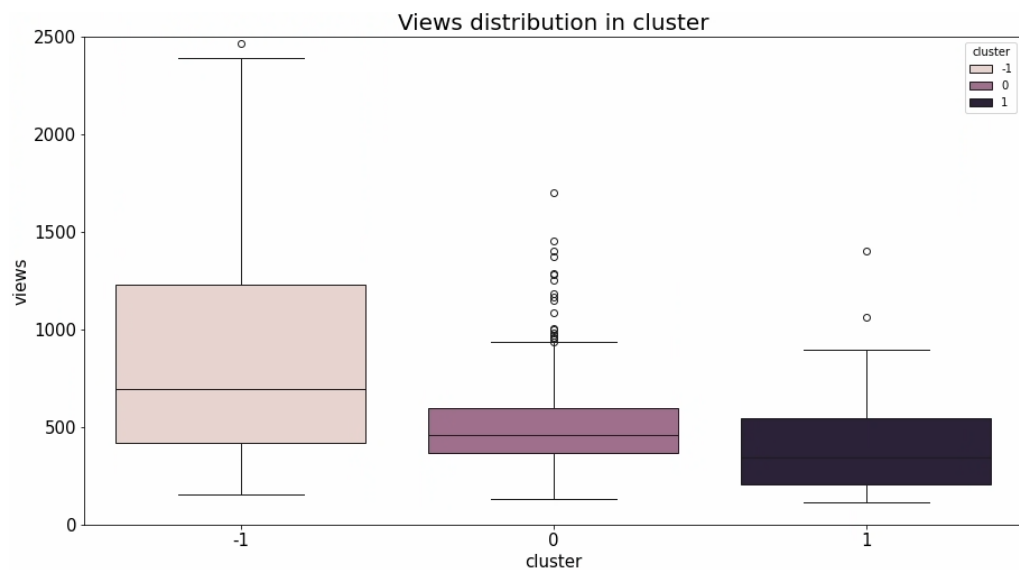


Рисунок 2.51 — Диаграмма размаха просмотров по кластерам

```
plt.figure(figsize=(15,8))
ax = sns.boxplot(data=posts_data, x='cluster', y='attachments', hue='cluster')
ax.tick_params(labelsize=15)
ax.set_xlabel('cluster', fontsize=15)
ax.set_ylabel('attachments', fontsize=15)
ax.set_title("Attachments distribution in cluster", fontsize=20)
ax.set_ylim([0, 10])
```

Рисунок 2.52 — Код построения диаграммы размаха фотографий по кластерам

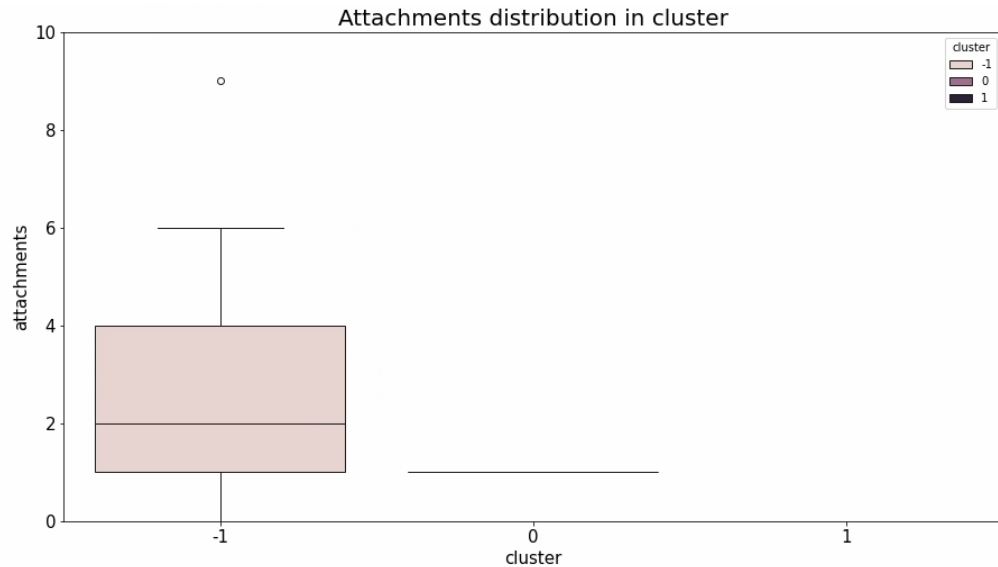


Рисунок 2.53 — Диаграмма размаха фотографий по кластерам

```
plt.figure(figsize=(15,8))
ax = sns.boxplot(data=posts_data, x='cluster', y='text_len', hue='cluster')
ax.tick_params(labelsize=15)
ax.set_xlabel('cluster', fontsize=15)
ax.set_ylabel('text length', fontsize=15)
ax.set_title("Text length distribution in cluster", fontsize=20)
ax.set_ylim([0, 300])
```

Рисунок 2.54 — Код построения диаграммы размаха длины текста по кластерам

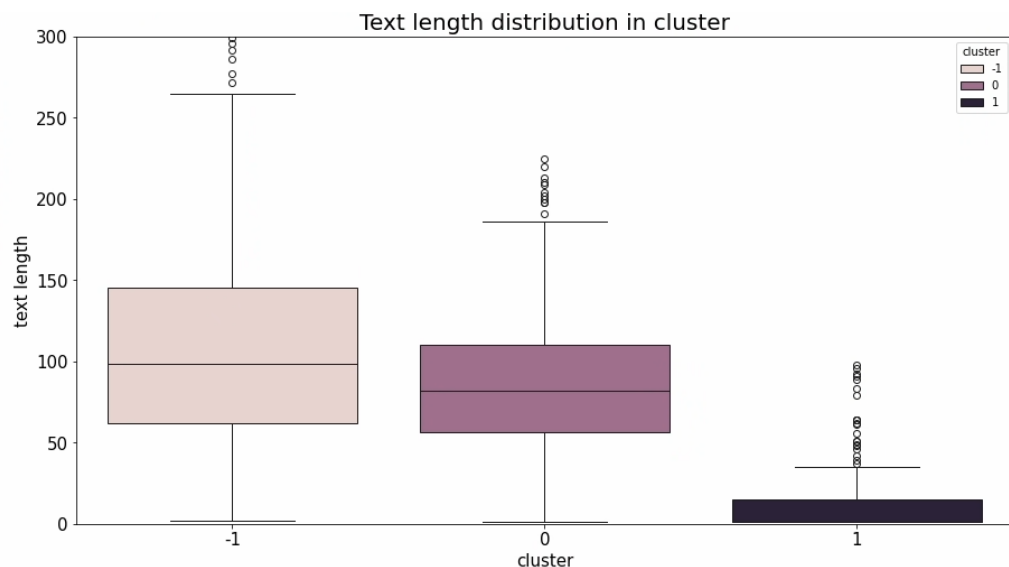


Рисунок 2.55 — Диаграмма размаха длины текста по кластерам

Из всех диаграмм размаха выше можно сделать вывод, что записи кластеризовались по количеству лайков, репостов, просмотров, фотографий и длины теста.

Выведем активность пользователей в группе по полу. Сразу исключим не активных пользователей, которые никогда не ставили лайки. По результатам можно заметить, что большая часть пользователей не проявляет активность в группе.

```
# activity in group by sex
plt.figure(figsize=(8, 8))
ax = sns.violinplot(x='sex', y='likes_number', data=info_filtered[info_filtered['likes_number'] < 100])
ax.set_xlabel('sex', fontsize=15)
ax.set_ylabel('likes', fontsize=15)
ax.tick_params(labelsize=15)
ax.set_title("Activity in group by sex", fontsize=20)
plt.show()
```

Рисунок 2.56 — Код для активности в группе по полу

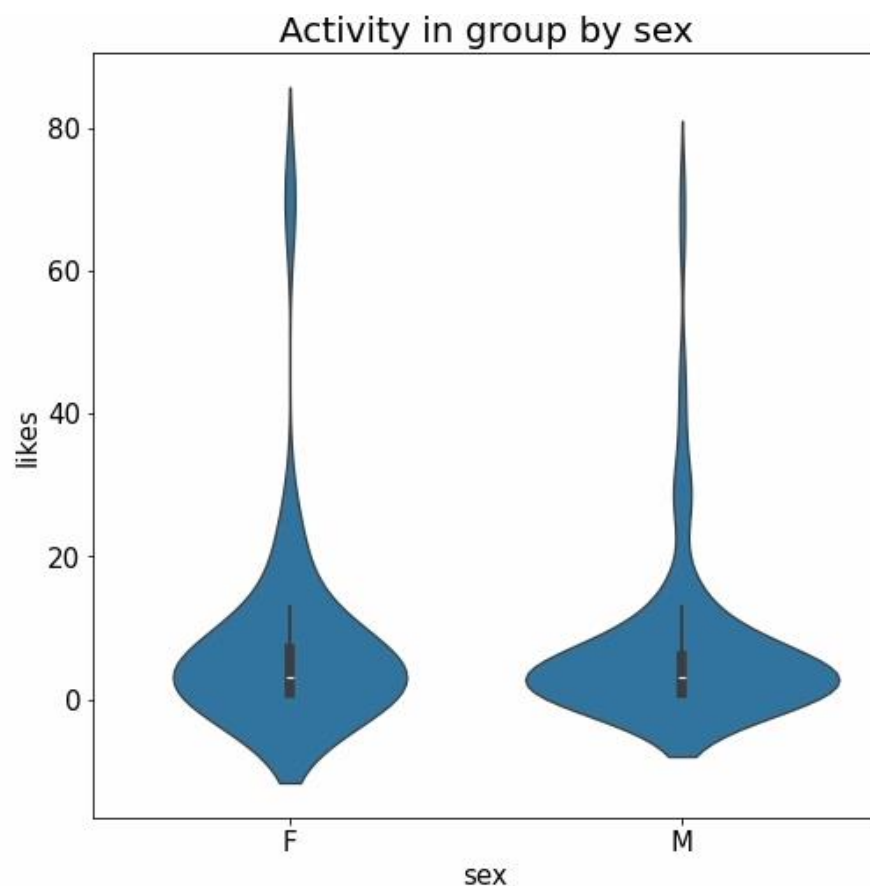


Рисунок 2.57 — Диаграмма активности в группе по полу

Теперь необходимо исследовать распределение полов в группе. Для этого нарисуем круговую диаграмму мужчин и женщин в группе. По результатам можно заметить, что мужчин в группе больше чем женщин.

```
# sex distribution
sex = usersDF_wlikes \
    .select('sex') \
    .rdd \
    .flatMap(lambda x: x) \
    .map(lambda x: (x, 1)) \
    .reduceByKey(lambda x, y: x + y) \
    .collect()

labels = [i[0] for i in sex]
amounts = [i[1] for i in sex]

fig, ax = plt.subplots(figsize=(12, 8))
ax.pie(amounts, labels=labels, autopct='%1.1f%%', textprops={'fontsize':14})
ax.set_title("Distribution by sex", fontsize=20)
plt.show()
```

Рисунок 2.58 — Код для получения распределения полов в группе

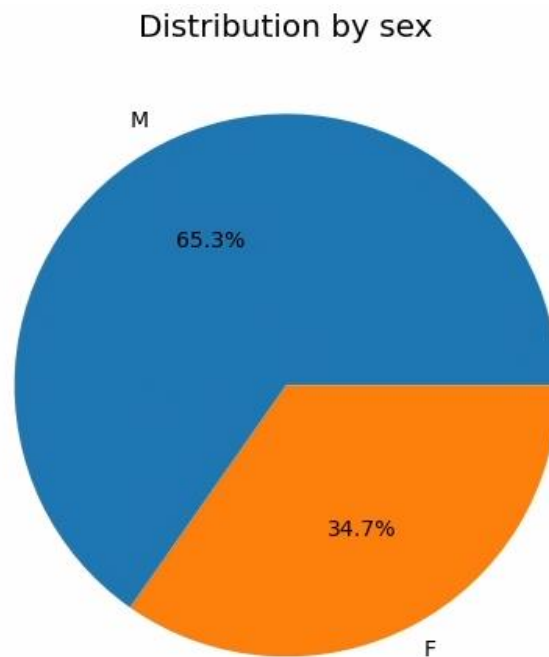


Рисунок 2.59 — Круговая диаграмма распределения полов в группе

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы, был реализован конвейер обработки данных используя различные инструменты, включая Hadoop, HDFS, Sqoop, Spark и Hive. Основной целью работы было провести качественный анализ данных о пользователях группы "Кафедра прикладной математики" в социальной сети ВК.

Для сбора информации о пользователях был написан скрипт на языке программирования python для взаимодействия с VK API.

Для обработки больших объемов данных о пользователях группы, были применены Hadoop и HDFS. Sqoop был использован для импорта данных из HDFS в реляционную базу данных для последующей обработки.

Анализ и визуализация данных были осуществлены, используя Spark и Hive.

Выполнение курсовой работы позволило приобрести навыки работы с различными инструментами обработки больших данных, а также провести качественный анализ данных о пользователях группы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Дзидзава Э.Т., Ахмедов К.М. БОЛЬШИЕ ДАННЫЕ И HADOOP: ОБЗОРНЫЙ ДОКЛАД // Вестник магистратуры. 2021. №1-1 (112).
2. Лыфарь Д. А. Обработка реляционных баз данных на графических процессорах // МСиМ. 2011. №1 (22).
3. М. С. Ефимова Интеллектуальный сбор информации из распределенных источников // Программные продукты и системы. 2019. №4.
4. Habr. Знакомство с Apache Spark. [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/companies/piter/articles/276675/>
5. Apache Spark. [Электронный ресурс]. – Режим доступа: <https://spark.apache.org/sql/>
6. Музалевский Д.С., Гапанюк Ю.Е. Пример проектного подхода к обучению в области обработки больших данных на основе построения рекомендательной системы с применением методов коллаборативной фильтрации с использованием Apache Spark и Python // Машиностроение и компьютерные технологии. 2016. №7.
7. Yandex cloud. Что такое Apache Kafka? Управляемые базы данных. [Электронный ресурс]. – Режим доступа: <https://cloud.yandex.ru/blog/posts/2021/02/managed-kafka-overview>
8. Big Data School. Apache Zookeeper. [Электронный ресурс]. – <https://www.bigdataschool.ru/wiki/zookeeper>