

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет прикладної математики  
Кафедра програмного забезпечення комп'ютерних систем

**КУРСОВИЙ ПРОЕКТ**

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: **Моніторингова система аналізу популярних відео сервісу  
YouTube**

**Студент**  
групи КП-61

**Свинарчук М. В.**

\_\_\_\_\_  
(підпис)

**Викладач**  
к.т.н, доцент кафедри  
СПіСКС

**Петрашенко А.В.**

\_\_\_\_\_  
(підпис)

Захищено з оцінкою \_\_\_\_\_

Київ – 2019

## АНОТАЦІЯ

Даний курсовий проект присвячений розробці вимог та створенню моніторингової система аналізу популярних відео сервісу YouTube. У документі викладено звітність процесу розробки програмного забезпечення системи.

Було досліджено актуальність та проблематику аналізу великих обсягів даних; проведено аналіз використаного інструментарію, а саме: обґрунтовано вибір мови програмування, бази даних та бібліотек; описано структуру бази даних та досліджено загальні можливості та переваги засобів масштабування, які надає обрана база даних; описано загальну структуру програмного забезпечення, його модулів та основних алгоритмів роботи; описано результати аналізу предметної галузі на основі графіків та текстової інформації.

Результатами даного проекту стала інтерактивна аналітика досліджуваних даних по кожній країні світу окремо чи по вибірці країн разом (див. Додаток А).

## ЗМІСТ

ВСТУП	4
1. АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ	5
2. СТРУКТУРА БАЗИ ДАНИХ	11
3. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	13
3.1. Загальна структура програмного забезпечення	13
3.2. Опис модулів програмного забезпечення	14
3.3. Опис основних алгоритмів роботи	19
4. АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ	21
5. ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ	41
ВИСНОВКИ	44
ЛІТЕРАТУРА	45
ДОДАТКИ	46
А. Графічні матеріали	46
Б. Фрагменти програмного коду	56

## ВСТУП

YouTube сьогодні є найбільшим веб-сайтом для хостингу різного жанру відео-контенту та однозначно тримає лідерську позицію як головна медіа платформа в світі. Загальна кількість користувачів платформи становить більше 1,3 млрд осіб. Кожну хвилину на YouTube завантажується 300 годин медіа контенту! Щодня проглядається майже 5 мільярдів відеороликів. YouTube отримує більше 30 мільйонів відвідувачів в день.

Отже, можна зробити висновок, що YouTube має глибокий вплив на суспільство цілого світу в усіх аспектах. Тому аналіз наборів даних даної платформи є досить інформативною та важливою задачею для компаній, що досліджують соціальні тенденції чи створюють прогнози майбутніх стратегічних бізнес-планів.

**Метою розробки даного курсового проекту є збір та фільтрація статистики популярних відео сервісу YouTube, проведення інтелектуального аналізу даних (data mining [1]) на її основі та формування бізнес-звітів з метою надання корисної інформації для рекламних агентств та інвесторів.**

Для потенційних клієнтів дана моніторингова система повинна допомогти знайти категорії відеоконтенту, які мають відносно високий бізнес-потенціал. Наприклад, можливі наступні критерії оцінювання: кількість відео в кожній категорії, коефіцієнт участі (кількість коментарів / кількість переглядів), рейтинг популярності (кількість вподобань / кількість переглядів), середній темп зростання в трендах, тощо. Маючи такого роду статистику, клієнту, будь то інвестор чи рекламне агенство, буде простіше знайти та визначитися яким саме категоріям відеоконтенту надати перевагу для власних інвестицій.

## 1. АНАЛІЗ ІНСТРУМЕНТАРІЮ ДЛЯ ВИКОНАННЯ КУРСОВОГО ПРОЕКТУ

### **Мова програмування – Python 3.7.2.**

Програмна частина курсового проекту була розроблена на мові програмування Python 3.7.2 [2]. Це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять Python привабливою мовою програмування для швидкої розробки програм, а також як засіб поєднання наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних переваг Python можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносимість програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- зручний для розв'язання математичних проблем;

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний

синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

На відміну від інших інтерпретованих мов програмування, Python активно використовується для проведення наукових розрахунків. В області аналізу та візуалізації даних, Python конкурує з багатьма предметно-орієнтованими мовами програмування та інструментами із відкритим вихідним кодом та із комерційними, а саме R, MATLAB, SAS, Stata та іншими.

### **СУБД – MongoDB 4.0**

MongoDB – документо-орієнтована система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СУБД, функціональними і зручними у формуванні запитів.

Основні можливості MongoDB:

- Документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних);
- Досить гнучка мова для формування запитів;
- Динамічні запити;
- Повна підтримка індексів;
- Профілювання запитів;
- Швидкі оновлення «на місці»;
- Ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео;
- Журналювання операцій, що модифікують дані в БД;

- Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардинг;
- Може працювати відповідно до парадигми MapReduce.

Вибір NoSQL [3] СУБД обґрунтовується наявністю великої кількості ненормалізованих даних, для яких необхідно забезпечити швидке збереження в СУБД завантаження з неї з подальшою їх обробкою. Надійність і цілісність даних в даному випадку не є головною задачею СУБД – втрата декількох записів при аналізі сотні тисяч об’єктів не вплине на результати аналізу.

При використанні реляційних баз даних також постає проблема вертикальної масштабованості — лавиноподібне зростання кількості даних в системі швидко вичерпує обчислювальні потужності заліза, які не можуть рости нескінченно. У такій ситуації хорошим виходом стане горизонтальне масштабування, коли кілька незалежних машин з’єднуються разом і кожна з них обробляє тільки частину запитів. Така архітектура робить можливим швидке збільшення потужності кластера шляхом додавання нового сервера. Розраховані на роботу в розподілених системах MongoDB сховища спочатку проектуються з таким розрахунком, що всі процедури реплікації, розподілу даних і забезпечення відмовостійкості виконуються самою СУБД. Тобто, ключові переваги MongoDB в розподілених системах полягають в процедурах шарингу і реплікації, які легко налаштовуються з утилітами даної СУБД (детальніше див. пункт 4. “Аналіз функціонування засобів масштабування”).

### **Бібліотеки:**

- *NumPy* – математична бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для

операцій з цими масивами. А також інструменти для інтеграції C/C++ і Fortran коду, корисну лінійну алгебру, перетворення Фур'є і можливості випадкових чисел. Крім очевидних наукових застосувань, NumPy також може бути використаний як ефективний багатовимірний контейнер загальних даних. Можна визначити довільні типи даних. Це дає змогу без проблем і швидкої інтеграції NumPy з різноманітними базами даних. NumPy має ліцензію за ліцензією BSD, що дозволяє повторно використовувати кілька обмежень [4];

- **Pandas** – бібліотека для обробки та аналізу даних. Pandas є open-source бібліотекою, ліцензована BSD (*Berkeley Source Distribution*), забезпечує швидкі, гнучкі та експресивні структури даних. Він прагне бути основним будівельним блоком високого рівня для здійснення практичного, реального аналізу даних у Python. Можливості бібліотеки: інструменти для обміну даними між структурами в пам'яті і файлами різних форматів, засоби поєднання даних і способи обробки відсутньої інформації, переформатування наборів даних, в тому числі створення зведених таблиць, зріз даних за значеннями індексу, розширені можливості індексування, вибірка з великих наборів даних, вставка і видалення стовпців даних, можливості угруповання дозволяють виконувати триетапні операції типу «поділ, зміна, об'єднання», злиття і об'єднання наборів даних [5];
- **NLTK** – Natural Language Toolkit, пакет бібліотек і програм для символної і статистичної обробки природної мови. NLTK є провідною платформою для побудови програм мовою Python з метою обробки даних природної мови. Пакет забезпечує зручні у використанні інтерфейси для більш ніж 50 корпусів і лексичних



ресурсів, таких як WordNet, поряд з набором бібліотек обробки тексту для класифікації, токенизації, стемінізації та інших інструментів для розв'язання задач комп'ютерної лінгвістики [6];

- **WordCloud** – бібліотека для візуального подання списку категорій (або тегів, також званих мітками, ярликами, ключовими словами, тощо). Зазвичай використовується для опису ключових слів (тегів) на веб-сайтах, або для представлення неформатованого тексту. Ключові слова найчастіше являють собою окремі слова, і важливість кожного ключового слова позначається розміром шрифту або кольором. Таке уявлення зручно для швидкого сприйняття найвідоміших термінів і для розподілу термінів за популярністю щодо один одного [7];
- **Matplotlib** – це бібліотека Python 2D, яка представляє числові дані у різноманітних форматах та інтерактивних середовищах на різних платформах. Matplotlib є гнучким, легко конфігурованим пакетом, який разом з NumPy, SciPy і IPython надає можливості, подібні до MATLAB. В даний час пакет працює з декількома графічними бібліотеками, включаючи wxWindows і PyGTK. Пакет підтримує багато видів графіків і діаграм: графіки (line plot), діаграми розсіювання (scatter plot), стовпчасті діаграми (bar chart) і гістограми (histogram), секторні діаграми (pie chart), діаграми «Стовбур-листя» (stem plot), контурні графіки (contour plot), поля градієнтів (quiver), спектральні діаграми (spectrogram) [8];
- **seaborn** – бібліотека візуалізації даних Python, яка базується на matplotlib. Забезпечує інтерфейс високого рівня для малювання привабливої та інформативної статистичної графіки. Деякі з функцій, які пропонує seaborn: 1) API, орієнтований на набір даних, для вивчення взаємозв'язків між кількома змінними;

2).спеціалізована підтримка використання категоріальних змінних для показу спостережень або сукупної статистики; 3) варіанти для візуалізації одновимірних або двовимірних розподілів і для порівняння їх між підмножинами даних; 4) автоматична оцінка та побудова лінійних регресійних моделей для різних залежних змінних; 5) зручні уявлення про загальну структуру складних наборів даних; 6) високоякісні абстракції для структурування багатосекторних сіток, що дозволяють легко створювати складні візуалізації; 7) інструменти для вибору кольорних палітр, які достовірно показують шаблони у ваших даних [9];

## 2. СТРУКТУРА БАЗИ ДАНИХ

Дані популярних відео сервісу YouTube були отримана шляхом регулярного скрапінгу сервісу за допомогою YouTube Data API. Інформація збиралася протягом довготривалого проміжку часу з конкретних країн світу (США, Великобританія, Німеччина, Канада, Франція та Індія), в результаті чого було сформовано датасет з 234544 записами. Також проект передбачає можливість самостійно збирати дані для аналізу.

В трендах YouTube по кожному відео було відібрано лише частину даних, з тих, що надає YouTube Data API. Ці несуть в собі корисну інформацію для майбутнього аналізу.

Структура бази даних складається лише зі схеми “videos”. Пропонується розглянути поля об’єкта “video”, який зберігається в даній таблиці:

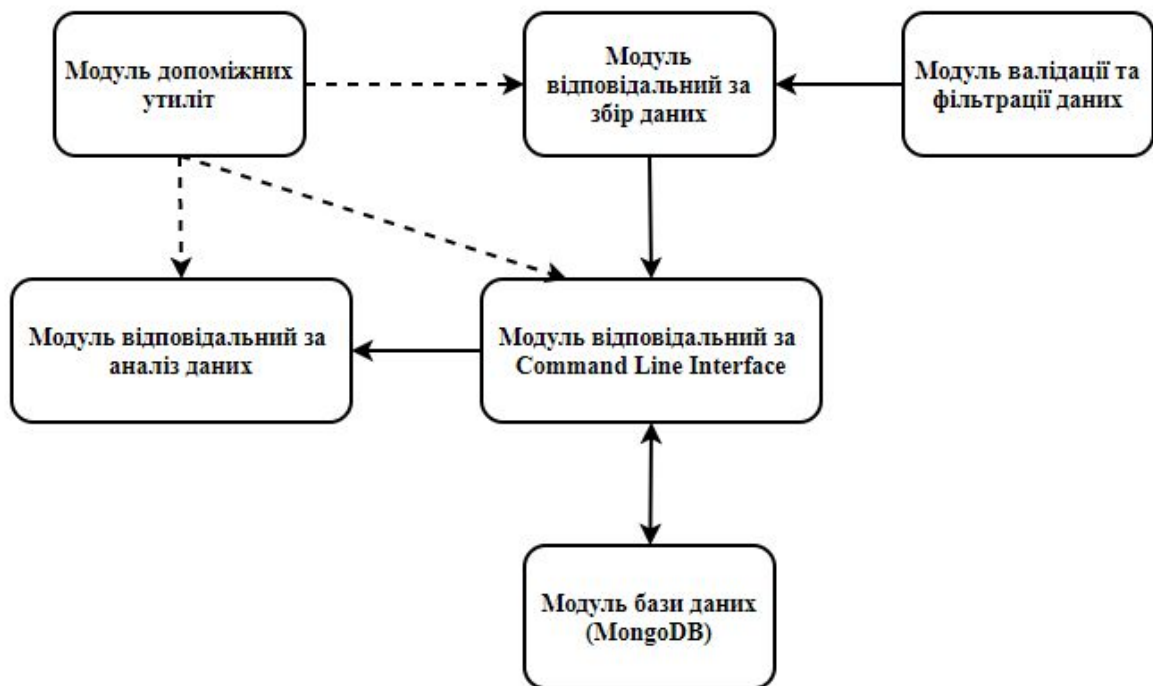
- **\_id** – ідентифікатор запису, унікальне поле у базі даних, первинний ключ. Тип даних ObjectId. Перед збереженням в базу даних дане поле формується на основі 3 груп даних відео: id відео в YouTube + trending\_date (дата скрапінгу, що є також датою, коли відео перебувало в трендах – Рік. Місяць. День) + country\_code (код країни для якої виконується скрапінг). Такий формат ідентифікатора запису реалізовано спеціально, щоб забезпечити унікальність відео в межах одного дня, тобто для попередження дублювання одних і тих же відео, а також щоб в майбутньому полегшити аналіз розподілу популярності відео по днях;
- **country\_code** – код країни для якої виконується скрапінг в форматі ISO 3166-1 alpha-2 (наприклад, US, UA, тощо). Тип даних String;
- **title** – назва відео. Тип даних String;

- **published\_at** – дата публікації відео на YouTube. Йдеться мова про дату, коли відео було опубліковане для перегляду на платформі, а не потрапило в тренди. Тип даних String;
- **channel\_title** – назва каналу, який опублікував конкретне відео. Тип даних String;
- **category** – жанр (категорія) відео, яку обрав власник для даного відео. Тип даних String;
- **trending\_date** – дата, коли відео перебувало в трендах. Формується автоматично системою при виконанні скрапінгу. Тип даних String;
- **tags** – ідентифікатор для категоризації, опису, пошуку даних і завдання внутрішньої структури. Тип даних String;
- **view\_count** – кількість переглядів відео. Тип даних Int32;
- **likes** – кількість вподобань відео. Дорівнює 0, якщо ratings\_disabled = true. Тип даних Int32;
- **dislikes** – кількість неприхильності відео. Дорівнює 0, якщо ratings\_disabled = true. Тип даних Int32;
- **comment\_count** – кількість коментарів до відео. Дорівнює 0, якщо comments\_disabled = true. Тип даних Int32;
- **thumbnail\_link** – посилання на зображення попереднього перегляду. Тип даних String;
- **comments\_disabled** – чи дозволено коментувати відео. Тип – Bool;
- **ratings\_disabled** – чи дозволено оцінювати відео. Тип – Bool;
- **description** – опис до відео. Тип даних String;

Також для пришвидшення пошуку відео по кодам країни (саме через поле country\_code знаходяться і завантажуються дані для подальшого аналізу) було створено додатковий індекс. Графічне представлення бази даних можна побачити на рис. 1-2 додатка А. Графічні матеріали.

### 3. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

#### 3.1. Загальна структура програмного забезпечення



*Рис. 3.1.1. Модульна організація програми*

Програмне забезпечення моніторингової системи аналізу популярних відео сервісу YouTube має мінімалістичний консольний інтерфейс. Тому структура програмного забезпечення значно спрощується порівняно з веб-інтерфейсом. Як видно з рис 3.1.1. загальна структура представляє собою модульну організацію, коли робота програми базується на взаємодії незалежних між собою частин (крім модуля відповідального за CLI, через який організована взаємодія решти модулів), які в свою чергу виконують конкретно визначений функціонал.

Всього структура даного програмного забезпечення складається з 16 python-файлів; файлу .env, який містить змінні оточення (в даному разі лише API\_KEY для YouTube Data API) та файлу country\_codes.txt в якому описані коди країн з яких сервер, відповідальний за збір даних, буде регулярно проводити скрапінг.

### 3.2. Опис модулів програмного забезпечення

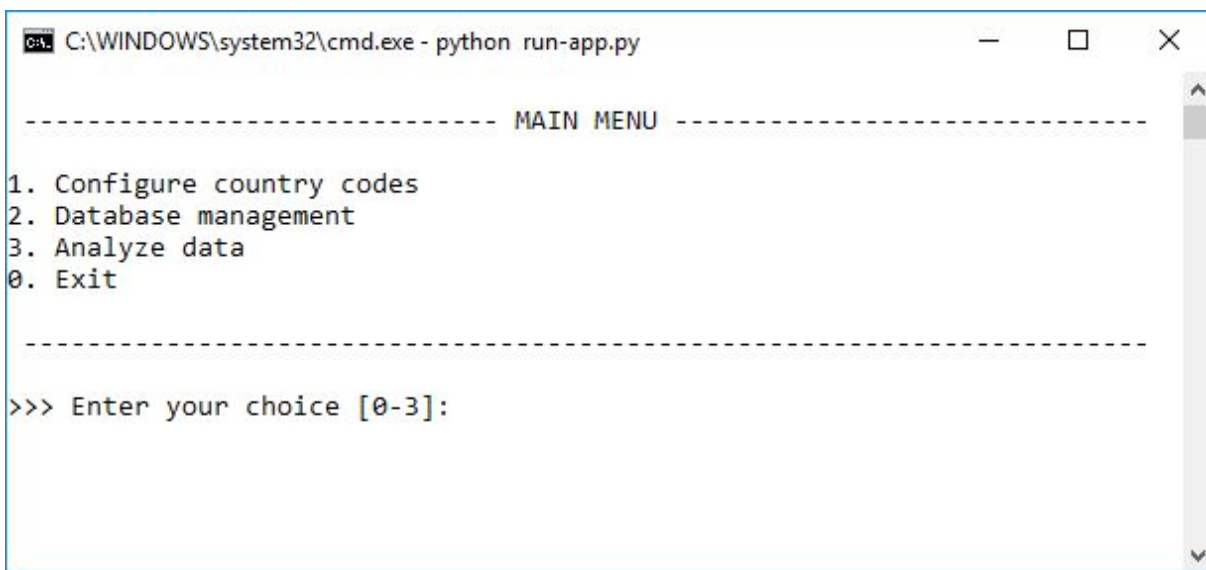
У програмі можна виділити 6 основних логічних модулів (рис. 3.1.1.), що мають певну самостійність і обмінюються один з одним даними. Модулям необов'язково відповідають конкретні класи чи ієрархії класів. Взаємодія між модулями відбувається за допомогою виклику методів. Пропонується розглянути кожен модуль окремо:

- **Модуль відповідальний за Command Line Interface**

Відповідальний за взаємодію з користувачем програми. Забезпечує зв'язність решти модулів та комутує дані між ними.

Всього в програмі налічується 4 меню для організації процесу аналізу популярних відео сервісу YouTube:

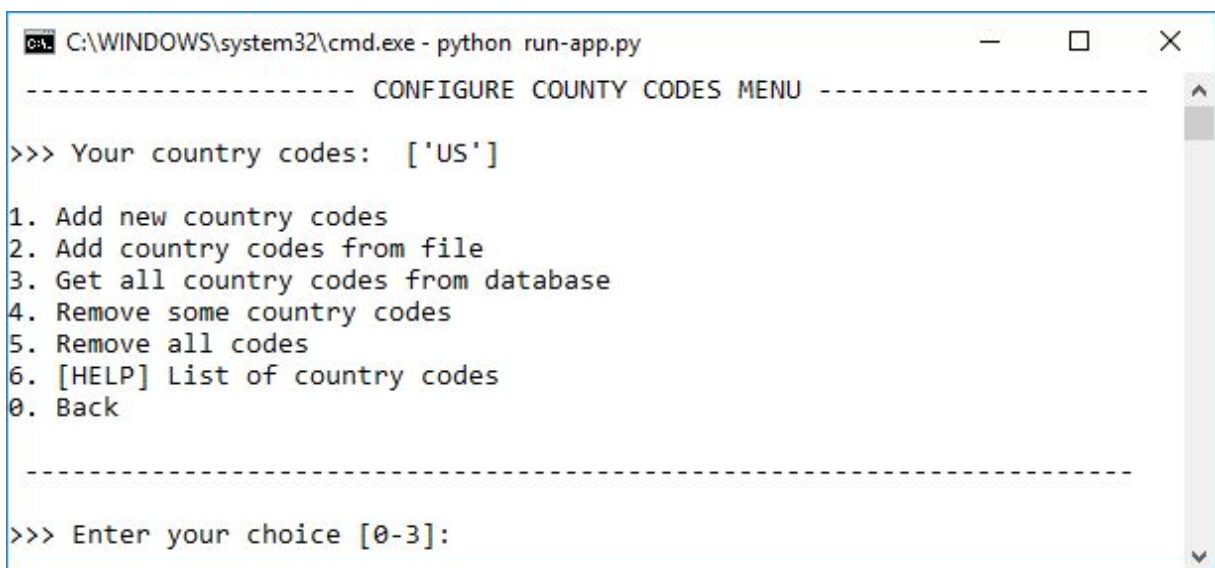
1. Головне меню – забезпечує переміщення між іншими меню інтерфейсу.



*Рис. 3.2.1. Головне меню програми*

2. Меню конфігурації кодів країн – дозволяє користувачу налаштувати з якими країнами він збирається працювати

(збирати дані чи аналізувати їх). Пункти меню: 1) власноруч вписати коди країн (система аналізує вхідні дані і не дозволить додати невалідні коди); 2) зчитати коди країн з файла (коди повинні бути записані окремо в нового рядку); 3) проаналізувати та отримати всі коди країн, дані яких вже наявні в базі; 4) виключно видалити; 5) видалити всі коди; 6) список існуючих кодів країн.



```
C:\WINDOWS\system32\cmd.exe - python run-app.py

----- CONFIGURE COUNTY CODES MENU -----

>>> Your country codes: ['US']

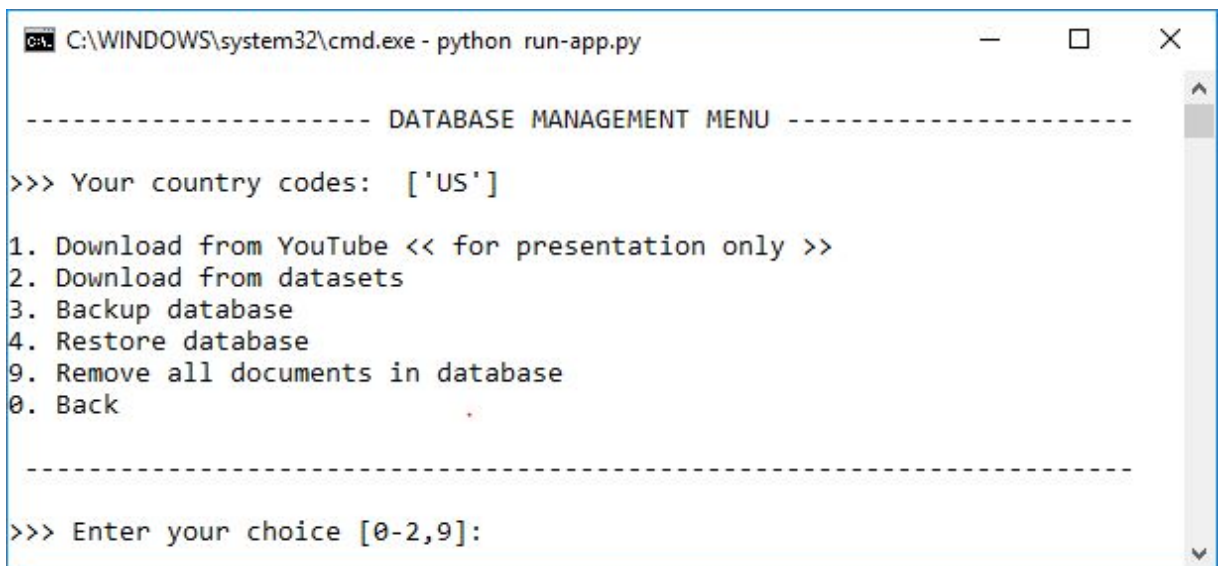
1. Add new country codes
2. Add country codes from file
3. Get all country codes from database
4. Remove some country codes
5. Remove all codes
6. [HELP] List of country codes
0. Back

-----

>>> Enter your choice [0-3]:
```

*Рис. 3.2.2. Меню конфігурації кодів країн*

3. Меню управління базою даних – дозволяє користувачу проводити маніпуляції з базою даних, а саме: завантажити дані з YouTube Data API по сконфігурованим раніше кодам країн, завантажити дані з наявних датасетів, зробити резервну копію або відновити базу даних (використовується стандартні утиліти MongoDB – mongodump та mongorestore), очистити базу даних.



```
C:\WINDOWS\system32\cmd.exe - python run-app.py

----- DATABASE MANAGEMENT MENU -----

>>> Your country codes: ['US']

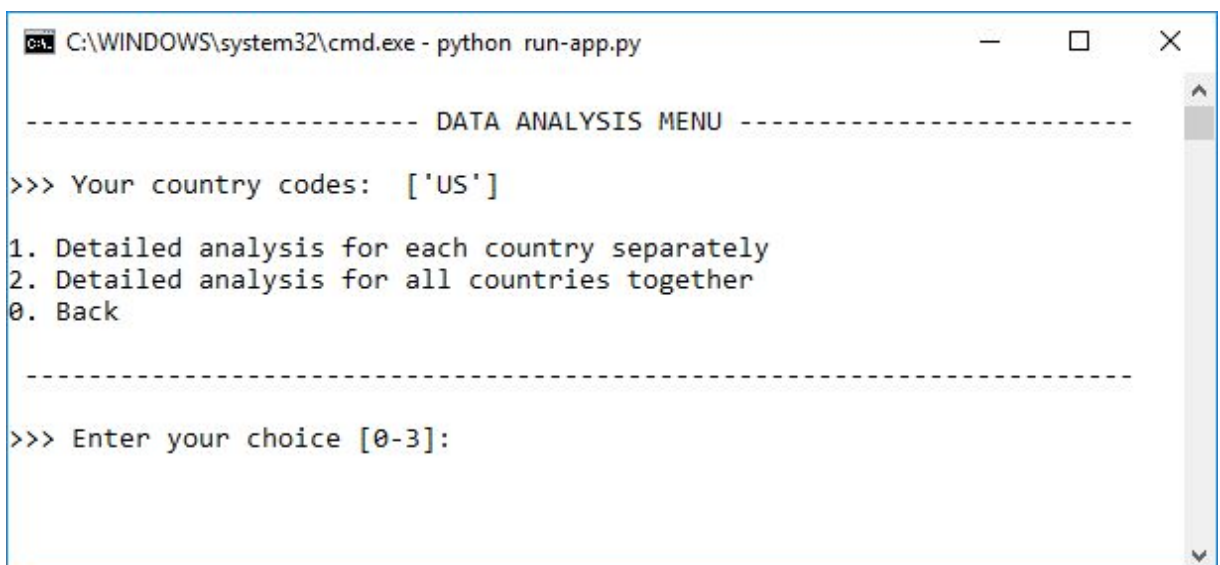
1. Download from YouTube << for presentation only >>
2. Download from datasets
3. Backup database
4. Restore database
9. Remove all documents in database
0. Back

-----

>>> Enter your choice [0-2,9]:
```

*Рис. 3.2.3. Меню управління базою даних*

4. Меню аналізу даних – дає можливість користувачу вибрати яким чином проводити аналіз: окремо по кожній країні зі списку чи разом. Після проведення аналізу програми відкриває директорію розташування результатів в Провіднику системи.



```
C:\WINDOWS\system32\cmd.exe - python run-app.py

----- DATA ANALYSIS MENU -----

>>> Your country codes: ['US']

1. Detailed analysis for each country separately
2. Detailed analysis for all countries together
0. Back

-----

>>> Enter your choice [0-3]:
```

*Рис. 3.2.4. Меню аналізу даних*



- **Модуль відповідальний за збір даних**

Відповідальний за збір даних для подальшого аналізу. Модуль реалізовує два способи збору даних - актуальних в цей час через YouTube Data API та зібраних раніше та представлених в якості готового датасету. Також модуль містить окрему утиліту, яка використовуючи можливості цього ж модулю дозволяє регулярно збирати дані (1 раз на добу) у встановлений користувачем час.

- **Модуль валідації та фільтрації даних**

Можливості модуля в основному використовуються при зборі даних, а саме для видалення надлишкових пробільних символів, для збору тегів відео в одну строку, для співставлення id категорії та власне жанра відео при зборі даних з YouTube, для підготовки даних у разі їх збереження в .csv формат, а також для форматування датасетів.

- **Модуль відповідальний за аналіз даних**

Містить перелік методів для аналізу даних. За допомогою бібліотек pandas, numpy, matplotlib, seaborn та WorldCloud будує відповідні графіки, хмари тегів, надає загальний тестовий аналіз. Детальніше з алгоритмами та результатами роботи можна ознайомитися в пункті 3.3. “Опис основних алгоритмів роботи” та в пункті 5. “Опис результатів аналізу предметної галузі”.

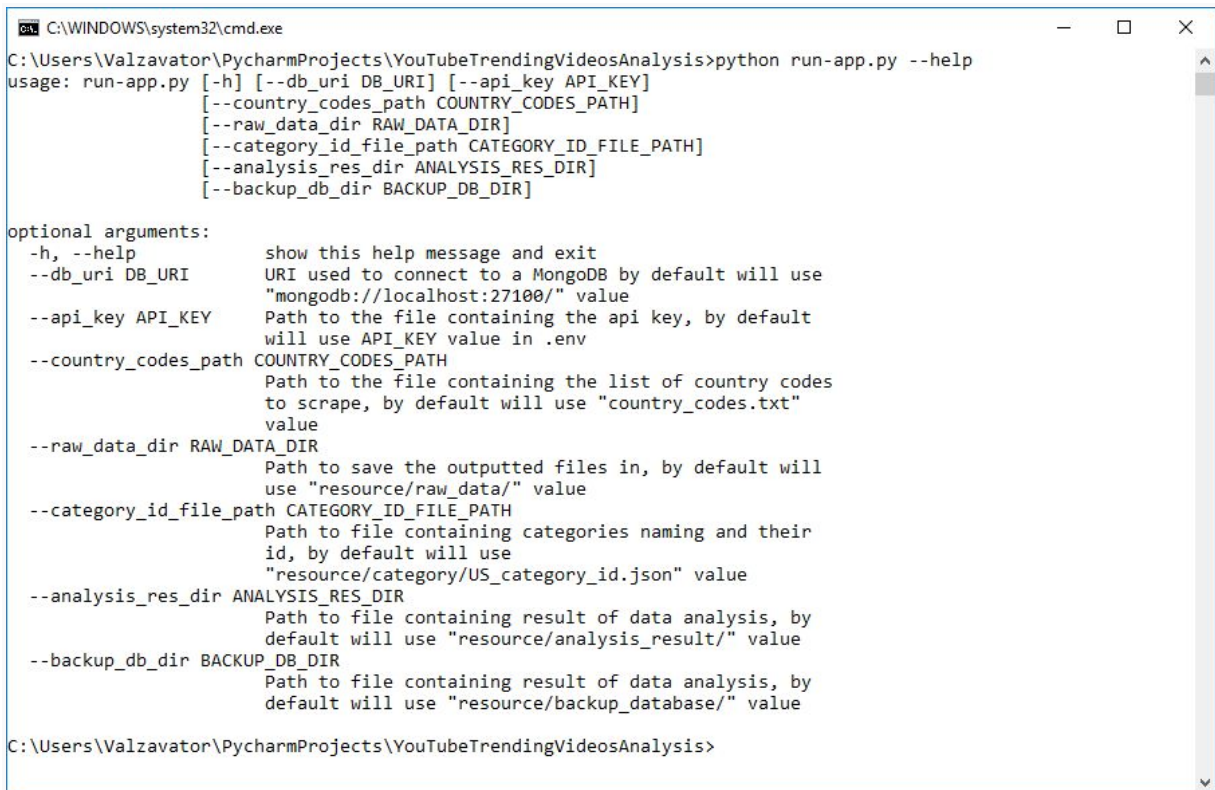
- **Модуль бази даних (MongoDB)**

Відповідальний за взаємодію програми та бази даних MongoDB. Реалізовує функції з записом, зчиткою та видаленням даних, видачі результатів пошуку по кодам країн, а також для створення резервної копії та відновлення бази даних.

- **Модуль допоміжних утиліт**

Містить функції зчитки даних з файлів (рядками та в форматі csv), а також запису даних (рядками чи в форматі csv). Також налаштовує

взаємодію з аргументами командного рядка, та задає глобальні змінні для всієї програми (db\_uri, api\_key, country\_codes\_path, raw\_data\_dir, category\_id\_file\_path, analysis\_res\_dir, backup\_db\_dir). Користувач може власноруч задати значення цих змінних при запуску програми в терміналі.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Valzavator\PycharmProjects\YouTubeTrendingVideosAnalysis>python run-app.py --help
usage: run-app.py [-h] [--db_uri DB_URI] [--api_key API_KEY]
                  [--country_codes_path COUNTRY_CODES_PATH]
                  [--raw_data_dir RAW_DATA_DIR]
                  [--category_id_file_path CATEGORY_ID_FILE_PATH]
                  [--analysis_res_dir ANALYSIS_RES_DIR]
                  [--backup_db_dir BACKUP_DB_DIR]

optional arguments:
  -h, --help            show this help message and exit
  --db_uri DB_URI        URI used to connect to a MongoDB by default will use
                        "mongodb://localhost:27100/" value
  --api_key API_KEY      Path to the file containing the api key, by default
                        will use API_KEY value in .env
  --country_codes_path COUNTRY_CODES_PATH
                        Path to the file containing the list of country codes
                        to scrape, by default will use "country_codes.txt"
                        value
  --raw_data_dir RAW_DATA_DIR
                        Path to save the outputted files in, by default will
                        use "resource/raw_data/" value
  --category_id_file_path CATEGORY_ID_FILE_PATH
                        Path to file containing categories naming and their
                        id, by default will use
                        "resource/category/US_category_id.json" value
  --analysis_res_dir ANALYSIS_RES_DIR
                        Path to file containing result of data analysis, by
                        default will use "resource/analysis_result/" value
  --backup_db_dir BACKUP_DB_DIR
                        Path to file containing result of data analysis, by
                        default will use "resource/backup_database/" value

C:\Users\Valzavator\PycharmProjects\YouTubeTrendingVideosAnalysis>
```

*Рис. 3.2.5. Можливі налаштування програми при її запуску в командній строці*

### 3.3. Опис основних алгоритмів роботи

Пропонується ознайомитися з основними алгоритмами програми, які забезпечують завантаження великого обсягу даних та його аналіз:

- **алгоритм зчитування даних з датасетів у форматі .csv (Додаток Б, лістинг 1)** – приймає на вхід перелік шляхів розміщення датасетів, за допомогою функції `read_csv()` з бібліотеки `pandas` зчитує дані з кожного файлу та повертає їх в форматі об'єктів `DataFrame`. Далі об'єднує дані всіх об'єктів в один та повертає їх як список словників;
- **алгоритм знаходження нормального розподілу кількості переглядів, уподобань, неприхильності та коментарів (Дод. Б, лістинг 2)** – розподіл ймовірностей випадкової величини, що характеризується густиною ймовірності;
- **алгоритм знаходження кореляції серед переглядів, уподобань, неприхильності та коментарів (Додаток Б, лістинг 3)** – виявляє чи існує істотна залежність однієї змінної від інших;
- **алгоритм знаходження кількості популярних відео по кожній категорії окремо (Додаток Б, лістинг 4)** – дають уявлення про те, наскільки популярною є кожна категорія на веб-сайті YouTube;
- **алгоритм знаходження розподілу переглядів, уподобань, неприхильності та коментарів у вигляді діаграми розмаху (Додаток Б, лістинг 5)** – відображає мінливість у вибірці статистичної сукупності, не роблячи ніяких припущень про базовий статистичний розподіл. Віддаль між різними частинами коробки вказує на ступінь дисперсії (розкиданості), асиметрію в даних і відображає викиди;
- **алгоритм знаходження розподілу середнього рівня переглядів, уподобань, неприхильності та коментарів у вигляді ламаної**

**кривої (Додаток Б, лістинг 6)** – показник обчислюється шляхом розподілу загальної кількості вимірюваної величини за кількістю відеороликів кожної категорії. Він описує середній рівень для різних категорій;

- **алгоритми знаходження середнього інтервалу в днях в якому відео перебувало в трендах у вигляді гістограми та графіка залежності середньої кількості днів які відео було в трендах від жанру відео (Додаток Б, лістинги 7)** – підраховує інтервал часу в якому відео залишається в трендах, віднявши дату розміщення в трендах з датою публікації;
- **алгоритми знаходження каналів по категоріям з найбільшою швидкістю зростання переглядів, уподобань, неприхильності та коментарів (Додаток Б, лістинг 8)** – знаходить топ-5 каналів по кожній категорії, чиї темпи зростання переглядів / уподобань / неприхильності / коментарів найбільші. Розраховується як кількість вимірюваної величини поділена на часовий інтервал в якому відео залишається в трендах;
- **алгоритм знаходження каналів чиї відео найчастіше потрапляють в тренди (Додаток Б, лістинг 9);**
- **алгоритм побудови хмари тегів для назв, опису та тегів відео (Додаток Б, лістинг 10)** – візуалізує найпоширеніші слова у назвах відео / опису / тегах. Чим більше вживане слово, тим більшим є його розмір шрифту;
- **алгоритм аналізу полярності настроїв по категоріям (Додаток Б, лістинг 11)** – використовуючи аналізатор настроїв від NLTK, вивчаємо полярності тегів відео по категоріям. Полярність в аналізі настроїв відноситься до виявлення орієнтації настроїв: позитивних, нейтральних і негативних.

#### 4. АНАЛІЗ ФУНКЦІОНУВАННЯ ЗАСОБІВ МАСШТАБУВАННЯ

Пропонується розглянути реплікацію та шардинг як техніки горизонтального масштабування баз даних. Також на прикладі бази даних курсового проекту проаналізуємо переваги та недоліки використання даних технологій.

**Реплікація** – одна з технік масштабування баз даних. Суть цієї техніки в тому, що дані з одного сервера бази даних постійно копіюються (реплікуються) на один або кілька других (звані репліками). Для додатка з'являється можливість використовувати не один сервер для обробки всіх запитів, а кілька (в MongoDB поки що не підтримується можливість читання даних з Secondary-реплік). Таким чином з'являється можливість розподілити навантаження з одного сервера на кілька [10].

У разі, коли Primary-сервер стає недоступним – система робить його Secondary, обирає новий Primary-сервер та перемикається на роботу з ним. А оскільки Secondary-репліки за нормальних умов містять повну копію Primary, то актуальність даних не буде втрачена.

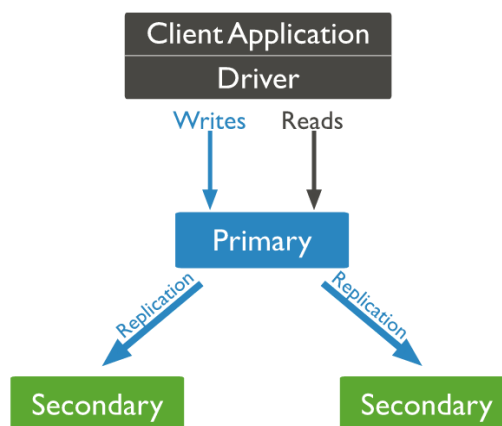


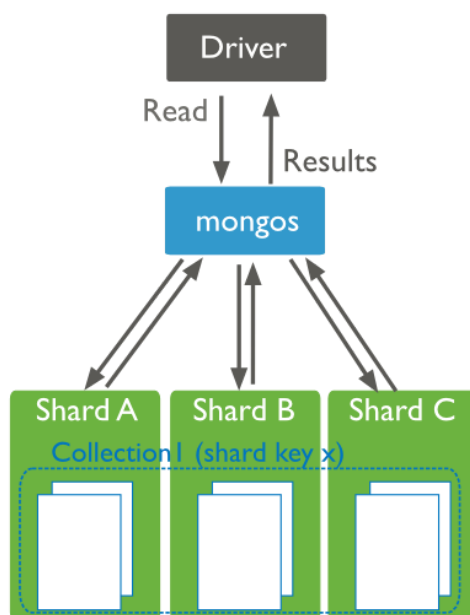
Рис. 4.1. Приклад реплікація бази даних з 3 репліками

**Шардинг** – це інша техніка масштабування роботи з даними. Суть її полягає в поділі бази даних на окремі частини так, щоб кожну з них можна було винести на окремий сервер [11].

Існують два види шардингу:

- Вертикальний шардинг – це виділення таблиці або групи таблиць на окремий сервер.
- Горизонтальний шардинг – це поділ однієї таблиці на різні сервера. Це необхідно використовувати для величезних таблиць, які не поміщаються на одному сервері. Поділ таблиці на частини робиться за таким принципом:
  - о На кількох серверах створюється одна і та ж таблиця (тільки структура, без даних).
  - о У додатку вибирається умова, за яким буде визначатися потрібне з'єднання (наприклад, парні на один сервер, а непарні - на інший).
  - о Перед кожним зверненням до таблиці відбувається вибір потрібного з'єднання.

В рамках курсового проекту нас цікавить саме горизонтальний шардинг.



*Рис. 4.2. Приклад горизонтального шардингу бази даних з 3 шардами*

Для прикладу горизонтального шардингу та реплікації створимо наступну схему організації бази даних MongoDB:

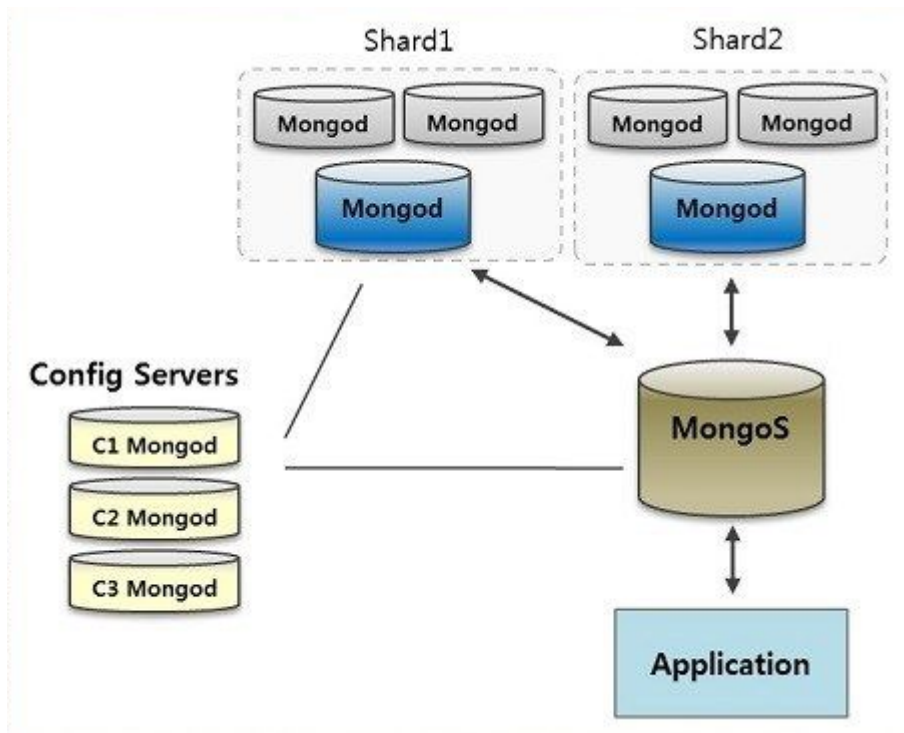


Рис. 4.3. Загальна схема шардинга в MongoDB

1. Створимо ReplicaSet “r0” для “Shard1” – запускаємо 3 сервера mongod та ініціалізуємо їх через mongo як ReplicaSet:

Запуск 3 серверів mongod (виконуємо кожну команду окремо в новому терміналі)

```
mongod.exe --replSet "r0" --port 27001 -dbpath=c:\data\db1\ --shardsvr
mongod.exe --replSet "r0" --port 27002 -dbpath=c:\data\db2\ --shardsvr
mongod.exe --replSet "r0" --port 27003 -dbpath=c:\data\db3\ --shardsvr
```

Налаштування ReplicaSet “r0” - приєднуємося до одного з серверів репліки через mongo та ініціалізуємо ReplicaSet

```
mongo.exe --port 27001
-----
> rs.initiate({"_id":"r0", members:[{"_id":0, host:"127.0.0.1:27001"}, {"_id":1,
host:"127.0.0.1:27002"}, {"_id":2, host:"127.0.0.1:27003"}]})
{
  "ok" : 1,
```

```

    "operationTime" : Timestamp(1558782883, 1),
    "$clusterTime" : {
      "clusterTime" : Timestamp(1558782883, 1),
      "signature" : {
        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAA="),
        "keyId" : NumberLong(0)
      }
    }
  }
}
r0:SECONDARY> rs.config()
{
  "_id" : "r0",
  "version" : 1,
  "protocolVersion" : NumberLong(1),
  "writeConcernMajorityJournalDefault" : true,
  "members" : [
    {
      "_id" : 0,
      "host" : "127.0.0.1:27001",
      "arbiterOnly" : false,
      "buildIndexes" : true,
      "hidden" : false,
      "priority" : 1,
      "tags" : {

      },
      "slaveDelay" : NumberLong(0),
      "votes" : 1
    },
    {
      "_id" : 1,
      "host" : "127.0.0.1:27002",
      "arbiterOnly" : false,
      "buildIndexes" : true,
      "hidden" : false,
      "priority" : 1,
      "tags" : {

      },
      "slaveDelay" : NumberLong(0),
      "votes" : 1
    },
    {
      "_id" : 2,
      "host" : "127.0.0.1:27003",
      "arbiterOnly" : false,
      "buildIndexes" : true,
      "hidden" : false,
      "priority" : 1,
      "tags" : {

```



```

        },
        "slaveDelay" : NumberLong(0),
        "votes" : 1
    }
],
"settings" : {
    "chainingAllowed" : true,
    "heartbeatIntervalMillis" : 2000,
    "heartbeatTimeoutSecs" : 10,
    "electionTimeoutMillis" : 10000,
    "catchUpTimeoutMillis" : -1,
    "catchUpTakeoverDelayMillis" : 30000,
    "getLastErrorModes" : {

    },
    "getLastErrorDefaults" : {
        "w" : 1,
        "wtimeout" : 0
    },
    "replicaSetId" : ObjectId("5ce923a347ad040789c93eb6")
}
}

```

2. Аналогічно створимо ReplicaSet “r1” для “Shard2” – запускаємо 3 сервера mongod та ініціалізуємо їх через mongo як ReplicaSet:

**Запуск 3 серверів mongod (виконуємо кожну команду окремо в новому терміналі)**

```

mongod.exe --replSet "r1" --port 27011 -dbpath=c:\data\db4\ --shardsvr
mongod.exe --replSet "r1" --port 27012 -dbpath=c:\data\db5\ --shardsvr
mongod.exe --replSet "r1" --port 27013 -dbpath=c:\data\db6\ --shardsvr

```

**Налаштування ReplicaSet “r1” - приєднуємося до одного з серверів репліки через mongo та ініціалізуємо ReplicaSet**

```

mongo.exe --port 27011
-----
> rs.initiate({"_id":"r1", members:[{"_id":0, host:"127.0.0.1:27011"}, {"_id":1,
host:"127.0.0.1:27012"}, {"_id":2, host:"127.0.0.1:27013"}]})
{
    "ok" : 1,
    "operationTime" : Timestamp(1558783228, 1),

```

```

        "$clusterTime" : {
            "clusterTime" : Timestamp(1558783228, 1),
            "signature" : {
                "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                "keyId" : NumberLong(0)
            }
        }
    }
}
r1:SECONDARY> rs.config()
{
    "_id" : "r1",
    "version" : 1,
    "protocolVersion" : NumberLong(1),
    "writeConcernMajorityJournalDefault" : true,
    "members" : [
        {
            "_id" : 0,
            "host" : "127.0.0.1:27011",
            "arbiterOnly" : false,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        },
        {
            "_id" : 1,
            "host" : "127.0.0.1:27012",
            "arbiterOnly" : false,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        },
        {
            "_id" : 2,
            "host" : "127.0.0.1:27013",
            "arbiterOnly" : false,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

```

```

        },
        "slaveDelay" : NumberLong(0),
        "votes" : 1
    }
],
"settings" : {
    "chainingAllowed" : true,
    "heartbeatIntervalMillis" : 2000,
    "heartbeatTimeoutSecs" : 10,
    "electionTimeoutMillis" : 10000,
    "catchUpTimeoutMillis" : -1,
    "catchUpTakeoverDelayMillis" : 30000,
    "getLastErrorModes" : {

    },
    "getLastErrorDefaults" : {
        "w" : 1,
        "wtimeout" : 0
    },
    "replicaSetId" : ObjectId("5ce924fb46afc3826fa90ac8")
}
}

```

3. Створюємо ReplicaSet “configReplSet” для серверів конфігурації – запускаємо 3 сервера mongod та ініціалізуємо їх через mongo як ReplicaSet:

**Запуск 3 серверів mongod (виконуємо кожну команду окремо в новому терміналі)**

```

mongod.exe --configsvr --replSet "configReplSet" --port 27111 -dbpath=c:\data\db7\
mongod.exe --configsvr --replSet "configReplSet" --port 27112 -dbpath=c:\data\db8\
mongod.exe --configsvr --replSet "configReplSet" --port 27113 -dbpath=c:\data\db9\

```

**Налаштування ReplicaSet “configReplSet” - приєднуємося до одного з серверів репліки через mongo та ініціалізуємо ReplicaSet**

```

mongo.exe --port 27111
-----
> rs.initiate({"_id":"configReplSet", members:[{"_id":0, host:"127.0.0.1:27111"},
{"_id":1, host:"127.0.0.1:27112"}, {"_id":2, host:"127.0.0.1:27113"}]})

```

```

{
  "ok" : 1,
  "operationTime" : Timestamp(1558784337, 1),
  "$gleStats" : {
    "lastOpTime" : Timestamp(1558784337, 1),
    "electionId" : ObjectId("000000000000000000000000")
  },
  "lastCommittedOpTime" : Timestamp(0, 0),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1558784337, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}

configReplSet:SECONDARY> rs.config()
{
  "_id" : "configReplSet",
  "version" : 1,
  "configsvr" : true,
  "protocolVersion" : NumberLong(1),
  "writeConcernMajorityJournalDefault" : true,
  "members" : [
    {
      "_id" : 0,
      "host" : "127.0.0.1:27111",
      "arbiterOnly" : false,
      "buildIndexes" : true,
      "hidden" : false,
      "priority" : 1,
      "tags" : {

      },
      "slaveDelay" : NumberLong(0),
      "votes" : 1
    },
    {
      "_id" : 1,
      "host" : "127.0.0.1:27112",
      "arbiterOnly" : false,
      "buildIndexes" : true,
      "hidden" : false,
      "priority" : 1,
      "tags" : {

      },
      "slaveDelay" : NumberLong(0),
      "votes" : 1
    },
  ],
}

```

```

        {
            "_id" : 2,
            "host" : "127.0.0.1:27113",
            "arbiterOnly" : false,
            "buildIndexes" : true,
            "hidden" : false,
            "priority" : 1,
            "tags" : {

            },
            "slaveDelay" : NumberLong(0),
            "votes" : 1
        }
    ],
    "settings" : {
        "chainingAllowed" : true,
        "heartbeatIntervalMillis" : 2000,
        "heartbeatTimeoutSecs" : 10,
        "electionTimeoutMillis" : 10000,
        "catchUpTimeoutMillis" : -1,
        "catchUpTakeoverDelayMillis" : 30000,
        "getLastErrorModes" : {

        },
        "getLastErrorDefaults" : {
            "w" : 1,
            "wtimeout" : 0
        },
        "replicaSetId" : ObjectId("5ce929518c1b2efd0a3d0f99")
    }
}

```

4. Зв'яжемо всі ініціалізовані репліки через mongos, ініціалізуємо шарди.

#### Запуск mongos з під'єднанням до нього репліки “configReplSet”

```

mongos.exe --configdb
configReplSet/127.0.0.1:27111,127.0.0.1:27112,127.0.0.1:27113 --port 27100

```

## Приєднуємося до mongos через mongo та ініціалізуємо 2 шарди з раніше створеними репліками

```
mongo.exe --port 27100

-----

mongos> sh.addShard("r0/127.0.0.1:27001,127.0.0.1:27002,127.0.0.1:27003")
{
  "shardAdded" : "r0",
  "ok" : 1,
  "operationTime" : Timestamp(1558784669, 1),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1558784669, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}

mongos> sh.addShard("r1/127.0.0.1:27011,127.0.0.1:27012,127.0.0.1:27013")
{
  "shardAdded" : "r1",
  "ok" : 1,
  "operationTime" : Timestamp(1558784684, 2),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1558784684, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

## Налаштовуємо шардинг для бази даних “videos\_analysis”

```
mongos> sh.enableSharding("videos_analysis")
{
  "ok" : 1,
  "operationTime" : Timestamp(1558826928, 2),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1558826928, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}

mongos> use vidoes_analysis
switched to db vidoes_analysis
```

5. Завантажуємо дані в базу даних за допомогою додатка та даємо команду на дозвіл шардингу.

Ініціалізуємо процес шардингу схеми “videos” в базі даних “videos\_analysis” з ключем “\_id”

```
mongos> sh.shardCollection("videos_analysis.videos", {_id: 1})
{
  "collectionsharded" : "videos_analysis.videos",
  "collectionUUID" : UUID("0bafcdab-18b1-49c4-8866-23ad951a2343"),
  "ok" : 1,
  "operationTime" : Timestamp(1558788049, 11),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1558788049, 11),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

Перевіряємо статус виконання шардингу на початку - 8 чанків все ще знаходяться на 1 шарді (r0 - 8)

```
mongos> db.printShardingStatus()
--- Sharding Status ---
...
shards:
{  "_id" : "r0",   "host" : "r0/127.0.0.1:27001,127.0.0.1:27002,127.0.0.1:27003",
  "state" : 1 }
{  "_id" : "r1",   "host" : "r1/127.0.0.1:27011,127.0.0.1:27012,127.0.0.1:27013",
  "state" : 1 }
active mongoses:
  "4.0.9" : 1
autosplit:
  Currently enabled: yes
balancer:
  Currently enabled:  yes
  Currently running:  yes
Collections with active migrations:
  videos_analysis.videos started at Sat May 25 2019 15:40:58 GMT+0300
(Oeieyiaey (ceia))
Failed balancer rounds in last 5 attempts:  0
Migration Results for the last 24 hours:
  No recent migrations
databases:
```

```

{ "_id" : "videos_analysis", "primary" : "r0", "partitioned" : true, "version"
: { "uuid" : UUID("d659539e-082f-4001-bd91-ceeb8ec29bc8"), "lastMod" : 1 } }
    videos_analysis.videos
        shard key: { "_id" : 1 }
        unique: false
        balancing: true
        chunks:
            r0      8
{ "_id" : { "$minKey" : 1 } } --> { "_id" : "7Kh18D9RVvw18.07.01GB" } on : r0
Timestamp(1, 0)
{ "_id" : "7Kh18D9RVvw18.07.01GB" } --> { "_id" : "FYg-xb5-rQU18.21.02DE" } on :
r0 Timestamp(1, 1)
{ "_id" : "FYg-xb5-rQU18.21.02DE" } --> { "_id" : "NdEBvZvd9Gs18.03.03FR" } on :
r0 Timestamp(1, 2)
{ "_id" : "NdEBvZvd9Gs18.03.03FR" } --> { "_id" : "VpsgERpY-y817.16.12DE" } on :
r0 Timestamp(1, 3)
{ "_id" : "VpsgERpY-y817.16.12DE" } --> { "_id" : "dH4ZXmeEJXs18.17.03US" } on :
r0 Timestamp(1, 4)
{ "_id" : "dH4ZXmeEJXs18.17.03US" } --> { "_id" : "lRkTgVynmbM18.05.03GB" } on :
r0 Timestamp(1, 5)
{ "_id" : "lRkTgVynmbM18.05.03GB" } --> { "_id" : "tH2U9_v7q9E18.22.04GB" } on :
r0 Timestamp(1, 6)
{ "_id" : "tH2U9_v7q9E18.22.04GB" } --> { "_id" : { "$maxKey" : 1 } } on : r0
Timestamp(1, 7)

```



## Перевіряємо статус виконання шардингу пізніше - 3 чанки вже були переміщені на 2 шард (r0 - 5, r0 - 3)

```
...
  shards:
{   "_id" : "r0",   "host" : "r0/127.0.0.1:27001,127.0.0.1:27002,127.0.0.1:27003",
"state" : 1 }
{   "_id" : "r1",   "host" : "r1/127.0.0.1:27011,127.0.0.1:27012,127.0.0.1:27013",
"state" : 1 }
  active mongoses: "4.0.9" : 1
  autosplit:
    Currently enabled: yes
  balancer:
    Currently enabled:  yes
    Currently running:  yes
    Collections with active migrations:
      videos_analysis.videos started at Sat May 25 2019 15:41:49 GMT+0300
(OeIeyiaey (ceia))
    Failed balancer rounds in last 5 attempts:  0
    Migration Results for the last 24 hours:
      3 : Success
  databases:
{   "_id" : "videos_analysis",   "primary" : "r0",   "partitioned" : true,   "version"
: {   "uuid" : UUID("d659539e-082f-4001-bd91-ceeb8ec29bc8"),   "lastMod" : 1 } }
    videos_analysis.videos
      shard key: { "_id" : 1 }
      unique: false
      balancing: true
      chunks:
          r0      5
          r1      3
{   "_id" : { "$minKey" : 1 } } -->> {   "_id" : "7Kh18D9RVvw18.07.01GB" } on : r1
Timestamp(2, 0)
{   "_id" : "7Kh18D9RVvw18.07.01GB" } -->> {   "_id" : "FYg-xb5-rQU18.21.02DE" } on :
r1 Timestamp(3, 0)
{   "_id" : "FYg-xb5-rQU18.21.02DE" } -->> {   "_id" : "NdEBvZvd9Gs18.03.03FR" } on :
r1 Timestamp(4, 0)
{   "_id" : "NdEBvZvd9Gs18.03.03FR" } -->> {   "_id" : "VpsgERpY-y817.16.12DE" } on :
r0 Timestamp(4, 1)
{   "_id" : "VpsgERpY-y817.16.12DE" } -->> {   "_id" : "dH4ZXmeEJXs18.17.03US" } on :
r0 Timestamp(1, 4)
{   "_id" : "dH4ZXmeEJXs18.17.03US" } -->> {   "_id" : "lRkTgVynmbM18.05.03GB" } on :
r0 Timestamp(1, 5)
{   "_id" : "lRkTgVynmbM18.05.03GB" } -->> {   "_id" : "tH2U9_v7q9E18.22.04GB" } on :
r0 Timestamp(1, 6)
{   "_id" : "tH2U9_v7q9E18.22.04GB" } -->> {   "_id" : { "$maxKey" : 1 } } on : r0
Timestamp(1, 7)
```

## Шардинг завершено - всі чанки збалансовано між шардами (r0 - 4, r0 - 4)

```
mongos> db.printShardingStatus()
--- Sharding Status ---
...
  shards:
{   "_id" : "r0",   "host" : "r0/127.0.0.1:27001,127.0.0.1:27002,127.0.0.1:27003",
"state" : 1 }
{   "_id" : "r1",   "host" : "r1/127.0.0.1:27011,127.0.0.1:27012,127.0.0.1:27013",
"state" : 1 }
  active mongoses:
      "4.0.9" : 1
  autosplit:
      Currently enabled: yes
  balancer: ...
      4 : Success
  databases:
{   "_id" : "videos_analysis", "primary" : "r0", "partitioned" : true, "version"
: {   "uuid" : UUID("d659539e-082f-4001-bd91-ceeb8ec29bc8"), "lastMod" : 1 } }
      videos_analysis.videos
          shard key: { "_id" : 1 }
          unique: false
          balancing: true
          chunks:
              r0      4
              r1      4
{   "_id" : { "$minKey" : 1 } } --> { "_id" : "7Kh18D9RVvw18.07.01GB" } on : r1
Timestamp(2, 0)
{   "_id" : "7Kh18D9RVvw18.07.01GB" } --> { "_id" : "FYg-xb5-rQU18.21.02DE" } on :
r1 Timestamp(3, 0)
{   "_id" : "FYg-xb5-rQU18.21.02DE" } --> { "_id" : "NdEBvZvd9Gs18.03.03FR" } on :
r1 Timestamp(4, 0)
{   "_id" : "NdEBvZvd9Gs18.03.03FR" } --> { "_id" : "VpsgERpY-y817.16.12DE" } on :
r1 Timestamp(5, 0)
{   "_id" : "VpsgERpY-y817.16.12DE" } --> { "_id" : "dH4ZXmeEJXs18.17.03US" } on :
r0 Timestamp(5, 1)
{   "_id" : "dH4ZXmeEJXs18.17.03US" } --> { "_id" : "lRkTgVynmbM18.05.03GB" } on :
r0 Timestamp(1, 5)
{   "_id" : "lRkTgVynmbM18.05.03GB" } --> { "_id" : "tH2U9_v7q9E18.22.04GB" } on :
r0 Timestamp(1, 6)
{   "_id" : "tH2U9_v7q9E18.22.04GB" } --> { "_id" : { "$maxKey" : 1 } } on : r0
Timestamp(1, 7)
```

## 6. Приклад відмовостійкості бази даних за допомогою техніки ReplicaSet:

Статус репліки "r0", коли всі репліки працюють. Primary - репліка "\_id" : 1. Secondary-репліки з "\_id" : 0 та 2 синхр. з даною реплікою.

```
r0:SECONDARY> rs.status()
{
  "set" : "r0",
  "date" : ISODate("2019-05-25T12:52:43.285Z"),
  "myState" : 2,
  "term" : NumberLong(6),
  "syncingTo" : "127.0.0.1:27002",
  "syncSourceHost" : "127.0.0.1:27002",
  "syncSourceId" : 1,
  "heartbeatIntervalMillis" : NumberLong(2000),
  "optimes" : {...},
  "lastStableCheckpointTimestamp" : Timestamp(1558788726, 1),
  "members" : [
    {
      "_id" : 0,
      "name" : "127.0.0.1:27001",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 1119,
      "optime" : {
        "ts" : Timestamp(1558788756, 1),
        "t" : NumberLong(6)
      },
      "optimeDate" : ISODate("2019-05-25T12:52:36Z"),
      "syncingTo" : "127.0.0.1:27002",
      "syncSourceHost" : "127.0.0.1:27002",
      ...
    },
    {
      "_id" : 1,
      "name" : "127.0.0.1:27002",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "uptime" : 1019,
      "optime" : {
        "ts" : Timestamp(1558788756, 1),
        "t" : NumberLong(6)
      },
    },
  ]
}
```

```

        "optimeDurable" : {
            "ts" : Timestamp(1558788756, 1),
            "t" : NumberLong(6)
        },
        "optimeDate" : ISODate("2019-05-25T12:52:36Z"),
        "optimeDurableDate" : ISODate("2019-05-25T12:52:36Z"),
        "lastHeartbeat" : ISODate("2019-05-25T12:52:41.596Z"),
        "lastHeartbeatRecv" : ISODate("2019-05-25T12:52:41.684Z"),
        "pingMs" : NumberLong(0),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "infoMessage" : "",
        "electionTime" : Timestamp(1558787754, 1),
        "electionDate" : ISODate("2019-05-25T12:35:54Z"),
        "configVersion" : 1
    },
    {
        "_id" : 2,
        "name" : "127.0.0.1:27003",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 1019,
        "optime" : {
            "ts" : Timestamp(1558788756, 1),
            "t" : NumberLong(6)
        },
        "optimeDurable" : {
            "ts" : Timestamp(1558788756, 1),
            "t" : NumberLong(6)
        },
        ...
        "syncingTo" : "127.0.0.1:27002",
        "syncSourceHost" : "127.0.0.1:27002",
        ...
    }
}, ...}

```

Статус репліки "r0", коли Primary-репліка з "\_id" : 1 стає недоступною - обирається нова Primary-репліка з "\_id" : 0, Secondary-репліка з "\_id" : 2 відтепер синхр. з даною реплікою.

```

r0:PRIMARY> rs.status()
{
    "set" : "r0",
    "date" : ISODate("2019-05-25T12:53:30.074Z"),

```

```

"myState" : 1,
"term" : NumberLong(7),
"syncingTo" : "",
"syncSourceHost" : "",
"syncSourceId" : -1,
"heartbeatIntervalMillis" : NumberLong(2000),
"optimes" : {...},
"lastStableCheckpointTimestamp" : Timestamp(1558788796, 1),
"members" : [
  {
    "_id" : 0,
    "name" : "127.0.0.1:27001",
    "health" : 1,
    "state" : 1,
    "stateStr" : "PRIMARY",
    "uptime" : 1166,
    "optime" : {
      "ts" : Timestamp(1558788808, 1),
      "t" : NumberLong(7)
    },
    "optimeDate" : ISODate("2019-05-25T12:53:28Z"),
    "syncingTo" : "",
    "syncSourceHost" : "",
    "syncSourceId" : -1,
    "infoMessage" : "",
    "electionTime" : Timestamp(1558788797, 1),
    "electionDate" : ISODate("2019-05-25T12:53:17Z"),
    "configVersion" : 1,
    "self" : true,
    "lastHeartbeatMessage" : ""
  },
  {
    "_id" : 1,
    "name" : "127.0.0.1:27002",
    "health" : 0,
    "state" : 8,
    "stateStr" : "(not reachable/healthy)",
    "uptime" : 0,
    "optime" : {
      "ts" : Timestamp(0, 0),
      "t" : NumberLong(-1)
    },
    "optimeDurable" : {
      "ts" : Timestamp(0, 0),
      "t" : NumberLong(-1)
    },
    "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
    "optimeDurableDate" : ISODate("1970-01-01T00:00:00Z"),
    "lastHeartbeat" : ISODate("2019-05-25T12:53:27.630Z"),
    "lastHeartbeatRecv" : ISODate("2019-05-25T12:53:18.195Z"),

```



```

"syncingTo" : "",
"syncSourceHost" : "",
"syncSourceId" : -1,
"heartbeatIntervalMillis" : NumberLong(2000),
"optimes" : {...},
"lastStableCheckpointTimestamp" : Timestamp(1558788796, 1),
"members" : [
  {
    "_id" : 0,
    "name" : "127.0.0.1:27001",
    "health" : 1,
    "state" : 1,
    "stateStr" : "PRIMARY",
    "uptime" : 1206,
    "optime" : {
      "ts" : Timestamp(1558788848, 1),
      "t" : NumberLong(7)
    },
    "optimeDate" : ISODate("2019-05-25T12:54:08Z"),
    "syncingTo" : "",
    "syncSourceHost" : "",
    "syncSourceId" : -1,
    "infoMessage" : "",
    "electionTime" : Timestamp(1558788797, 1),
    "electionDate" : ISODate("2019-05-25T12:53:17Z"),
    "configVersion" : 1,
    "self" : true,
    "lastHeartbeatMessage" : ""
  },
  {
    "_id" : 1,
    "name" : "127.0.0.1:27002",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 4,
    "optime" : {
      "ts" : Timestamp(1558788848, 1),
      "t" : NumberLong(7)
    },
    "optimeDurable" : {
      "ts" : Timestamp(1558788848, 1),
      "t" : NumberLong(7)
    },
    ...
    "syncingTo" : "127.0.0.1:27003",
    "syncSourceHost" : "127.0.0.1:27003",
    ...
  },
  {

```

```

        "_id" : 2,
        "name" : "127.0.0.1:27003",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 1106,
        "optime" : {
            "ts" : Timestamp(1558788848, 1),
            "t" : NumberLong(7)
        },
        "optimeDurable" : {
            "ts" : Timestamp(1558788848, 1),
            "t" : NumberLong(7)
        },
        ...
        "syncingTo" : "127.0.0.1:27001",
        "syncSourceHost" : "127.0.0.1:27001",
        ...
    },
    ...}

```

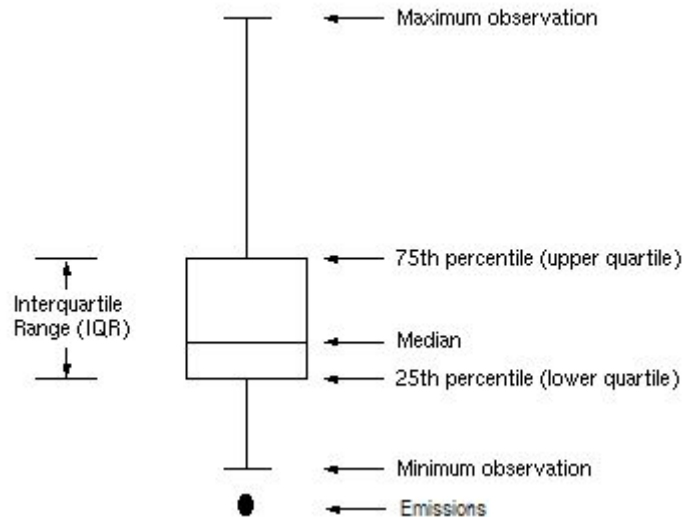


## 5. ОПИС РЕЗУЛЬТАТІВ АНАЛІЗУ ПРЕДМЕТНОЇ ГАЛУЗІ

У додатку А “Графічні матеріали” представлено 17 графіків (див. рис. 3-19) демонстрації результатів аналізу моніторингової системи аналізу популярних відео сервісу YouTube по США. Пропонуємо детальніше ознайомитися з результатами:

- На рис. 3 представлено графік нормального розподілу кількості переглядів, вподобань, неприхильності та коментарів. За ним можна переконатися у “адекватності” статистичних даних та визначити моду, медіану і середнє значення натурального логарифму вимірюваної величини (відповідає  $X$  координаті при найбільшому значенні  $Y$ );
- На рис. 4 представлено графік кореляції між кількістю переглядів, вподобань, неприхильності та коментарів. Кореляція показує зв’язок між двома кількісними змінними. Коефіцієнт кореляції показує ступінь, до якого дві змінні пов’язані (наскільки спільно чи подібно змінюються їх значення для різних спостережень). Навіть якщо дві змінні виглядають пов’язаними між собою, це не значить, що одна спричинила іншу. Так, можна побачити, що кількість вподобань і коментарів більше зростають з кількістю переглядів, чим неприхильність для аналізованих відео;
- На рис. 5 представлено графік співвідношення кількості відео в трендів по категоріям, де можна побачити, що більше половини контенту серед трендів складають відео з категорій Розваги та Музика;
- На рис. 6 - 9 представлено діаграма розмаху розподілу переглядів, вподобань, неприхильності та коментарів до відео по категоріям. Позначення на діаграмах мають наступні значення (див рис. 5.1):
  - Мінімальне значення в спостереженні;

- о Перший (або нижній) кuartиль;
- о Медіана (серединне значення);
- о Третій (або верхній) кuartиль;
- о Максимальне значення;
- о Викиди (точки поза макс. та мін. значеннями).



*Рис. 5.1. Позначення на діаграмі розмаху*

- На рис. 10 - 13 представлено графіки середнього рівня кількості переглядів, вподобань, неприхильності та коментарів у вигляді ламаної кривої. Вони менш інформативні ніж, наприклад, діаграми розмаху, але є більш інтуїтивно-зрозумілими для перегляду. Так можна побачити, що за кількістю переглядів категорія Музика значно випереджає інші, за вподобаннями - категорії Музика та Некомерційна діяльність, а за неприхильністю та кількістю коментарів - лише Некомерційна діяльність. Але тут криється проблема аналізу лише за середнім значенням кількості вимірюваної величини. Якщо звернутися до графіка співвідношення кількості відео в трендів по категоріям, то

можна побачити, що Некомерційна діяльність зовсім не популярний жанр, також це видно з діаграми розмаху;

- На рис. 14 представлено гістограму середнього інтервалу в якому відео було в трендах. Як можемо бачити, в середньому відео перебувають в трендах від 1 до 7 днів;
- На рис. 15 представлено графік залежності кількості днів які відео було в трендах від категорії відео. Так, відео з категорії Шоу найдовше тримаються в трендах (до 11 днів);
- На рис. 16 - 18 представлено хмари тегів з найпопулярнішими словосполученнями серед тегів, назв відео та опису до відео;
- На рис. 19 представлено графік полярності настроїв відео по категоріям. Так, можна побачити, що в категоріях Новини і політика, Комедійні шоу та Ігри більше заперечень та негативної тональності у порівнянні з категоріями Стиль, Подорожі та Блоги, де переважає позитивна тональність.

## ВИСНОВКИ

Моніторингова система аналізу популярних відео сервісу YouTube – це проект, основною задачею якого є дослідження соціальних тенденцій та створення бізнес–статистики з метою виокремити потенційно успішні категорії відео для рекламних агентств та інвесторів.

Для виконання курсового проекту була опрацьована відповідна технічна література (див. пункт “Література”). В ході розробки програмного забезпечення створено утиліту для безперервного збору статистичних даних та засоби для отримання аналогічних даних з готових датасетів чи її зберігання. Для аналізу тільки корисної статистики було розроблено алгоритми валідації та фільтрації інформації, які ще на етапі отримання даних відсіюють невалідні записи. За допомогою існуючих математичних та графічних пакетів мови Python система аналізує статистичні дані та генерує інтуїтивно зрозумілі звіти у вигляді графіків, гістограм, хмар тегів та текстової інформації. Причому аналіз можна проводити по будь-якій країні світу, де YouTube знаходиться у відкритому доступі, чи взагалі по виборці країн, якщо користувача цікавить статистика по регіону (наприклад, СНГ, ЄС, тощо).

Було досліджено основні переваги та недоліки використання NoSQL бази даних. Також освоєно та протестовано технології горизонтального масштабування, резервування та відновлення бази даних.

В ході виконання даного курсового проекту було досягнуто поставленої мети – набути практичні навички розробки сучасного програмного забезпечення, адміністрування NoSQL бази даних, а також навички оформлення відповідного текстового, програмного та графічного матеріалу у формі проектної документації.

## ЛІТЕРАТУРА

1. Data mining [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: [https://ru.wikipedia.org/wiki/Data\\_mining](https://ru.wikipedia.org/wiki/Data_mining);
2. Python [Електронний ресурс]. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Python>;
3. Преимущества и недостатки нереляционных баз данных [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://veesp.com/ru/blog/sql-or-nosql>;
4. NumPy [Електронний ресурс]. – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/NumPy>;
5. Pandas [Електронний ресурс]. – Режим доступу до ресурсу: <https://ru.wikipedia.org/wiki/Pandas>;
6. Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python. — O'Reilly Media, 2009.
7. Tag cloud [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Tag\\_cloud](https://en.wikipedia.org/wiki/Tag_cloud);
8. Matplotlib [Електронний ресурс]. – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Matplotlib>;
9. An introduction to seaborn [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <http://seaborn.pydata.org/introduction.html>;
10. Репликация данных [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://bit.ly/2WnlHz7>;
11. Шардинг и репликация [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://bit.ly/2vAYScb>.

## ДОДАТКИ

### А. Графічні матеріали


Key	Type
 _id	String
 trending_date	String
 title	String
 channel_title	String
 published_at	String
 tags	String
 view_count	Int32
 likes	Int32
 dislikes	Int32
 comment_count	Int32
 thumbnail_link	String
 comments_disabled	Bool
 ratings_disabled	Bool
 description	String
 category	String
 country_code	String

Рис. 1. Схема об'єктів “videos”, що зберігаються в базі даних MongoDB

```
> db.videos.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "videos_analysis.videos"
  },
  {
    "v" : 2,
    "key" : {
      "country_code" : -1
    },
    "name" : "country_code_-1",
    "ns" : "videos_analysis.videos"
  }
]
```

Рис. 2. Індеси для схеми “videos”

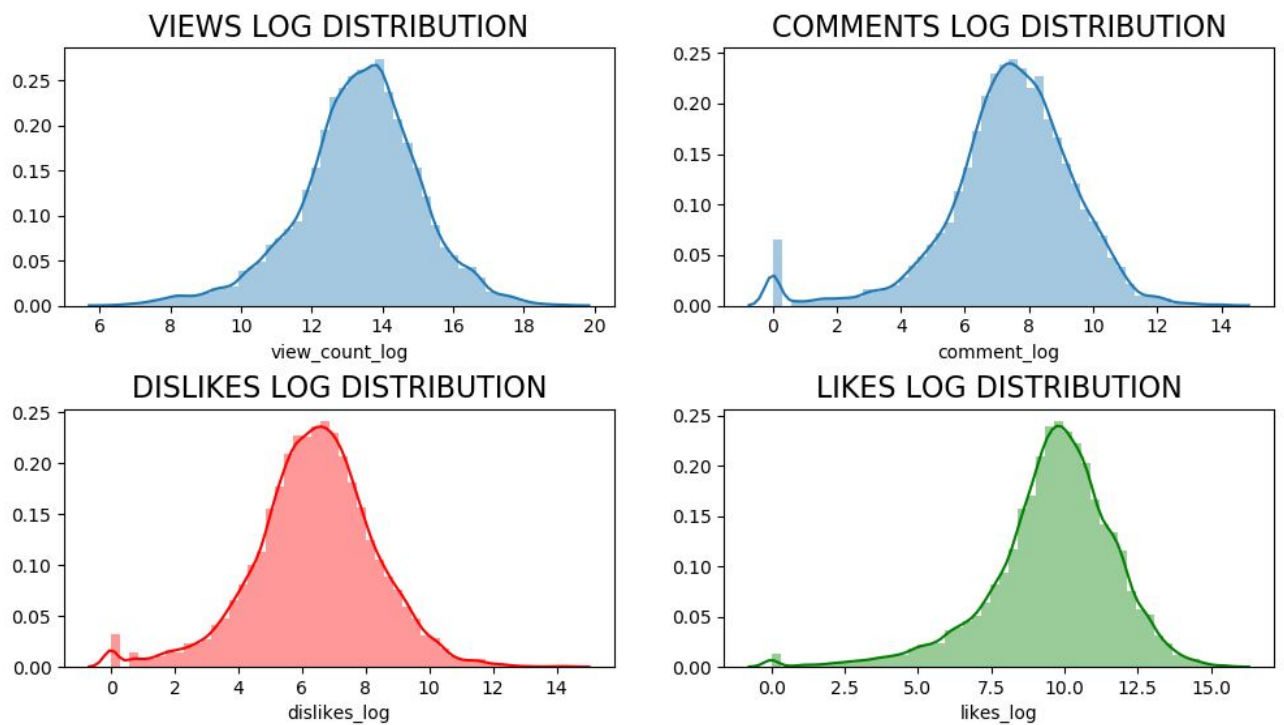


Рис. 3. Нормальний розподіл кількості переглядів, вподобань, неприхильності та коментарів

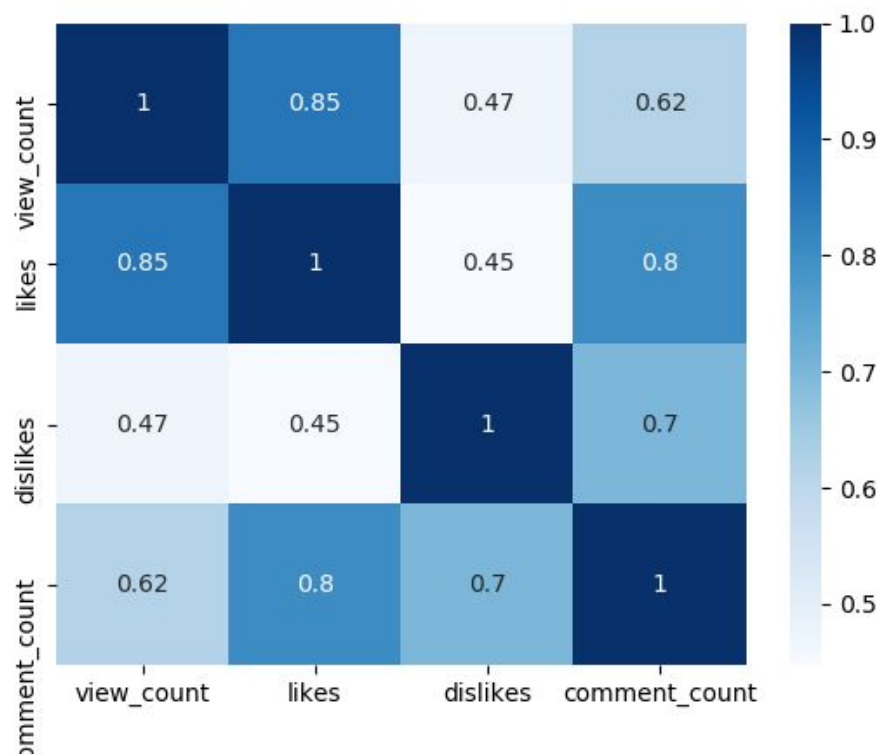


Рис. 4. Графік кореляції між кількістю переглядів, вподобань, неприхильності та коментарів

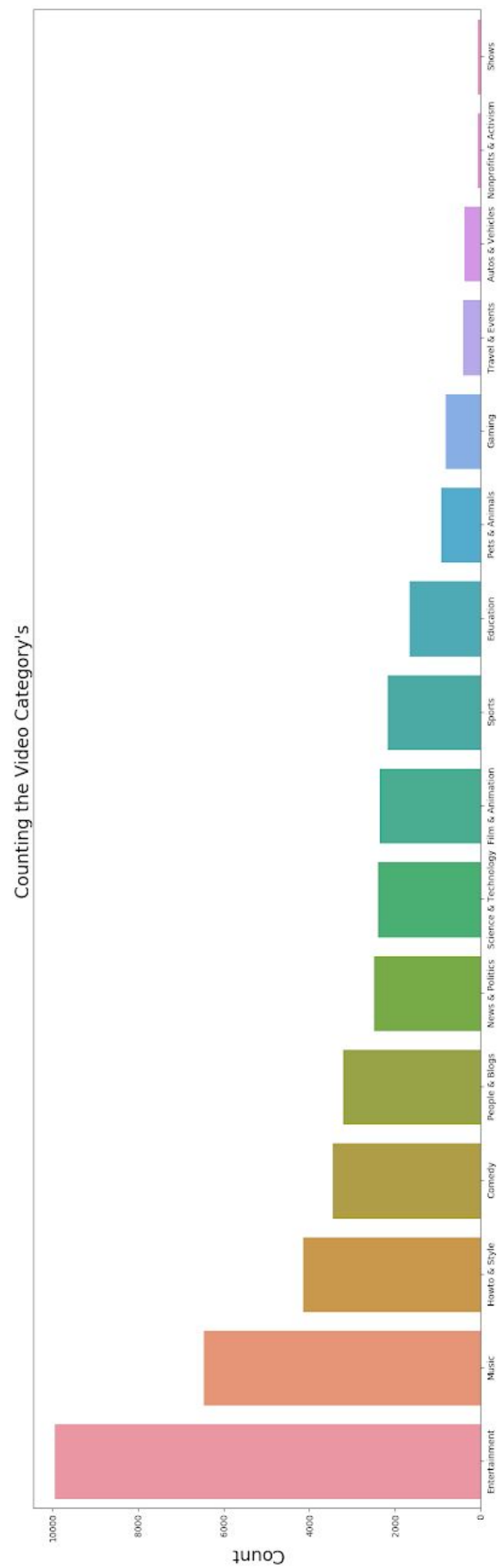


Рис. 5. Графік співвідношення кількості відео в трендів по категоріям



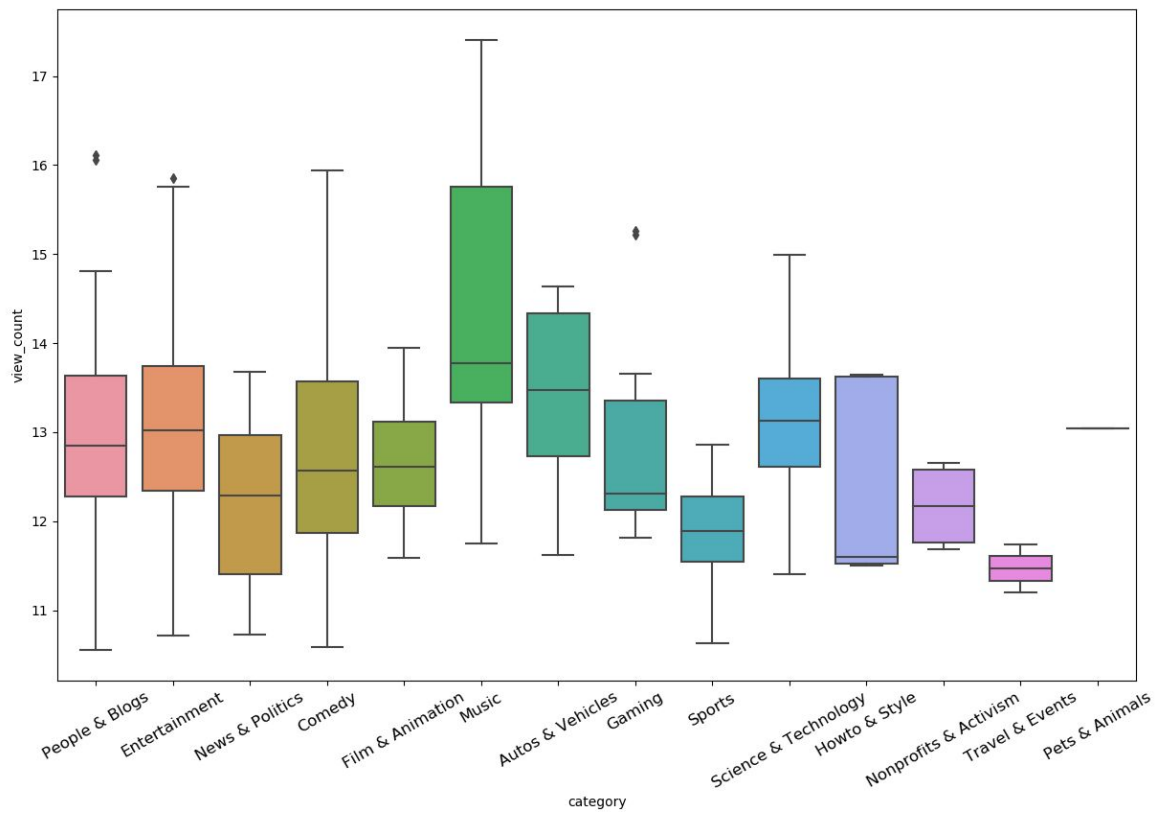


Рис. 6. Діаграма розмаху розподілу переглядів відео по категоріям

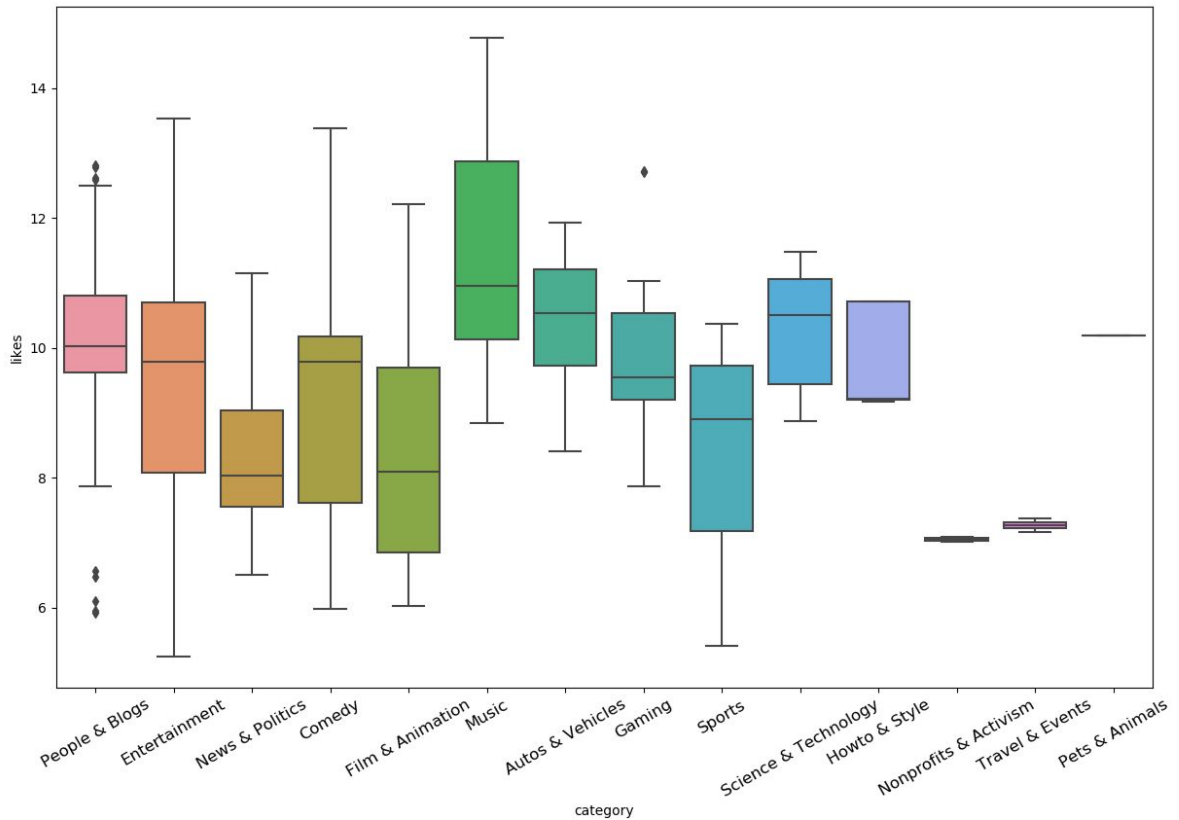


Рис. 7. Діаграма розмаху розподілу вподобань відео по категоріям

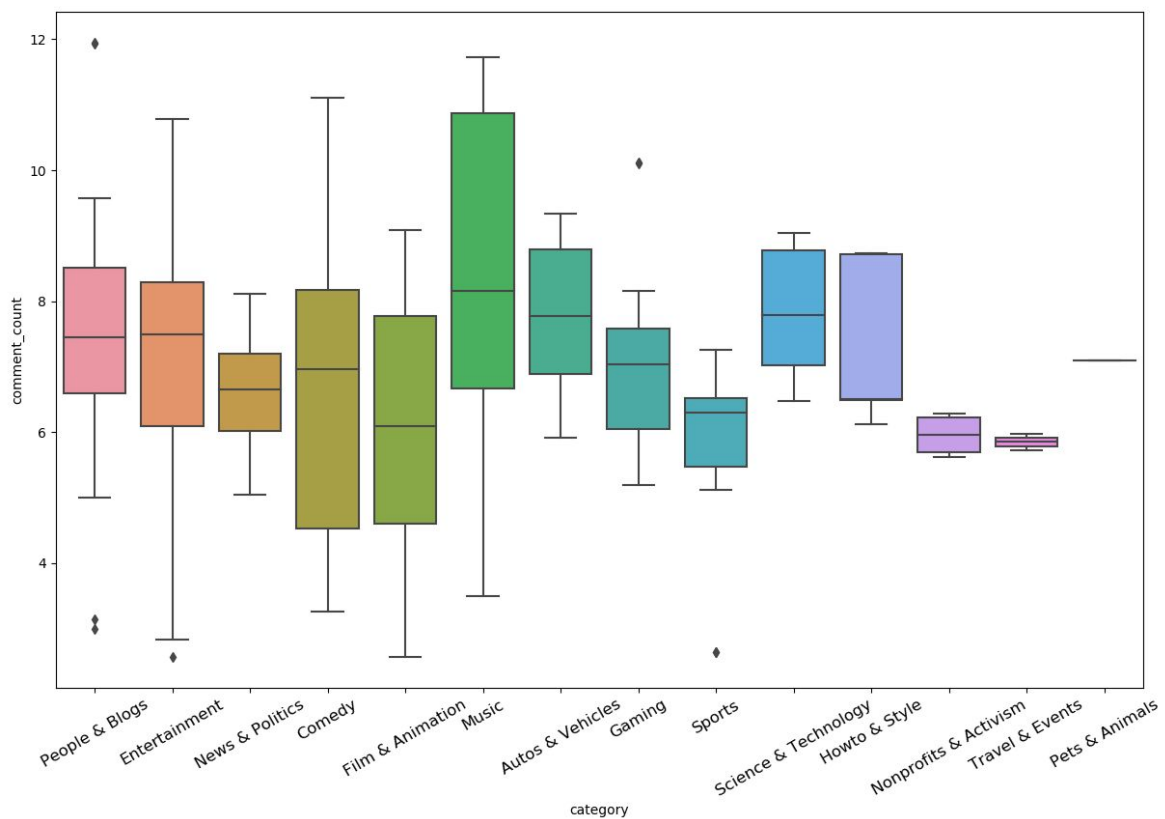


Рис. 8. Діаграма розмаху розподілу коментарів до відео по категоріям

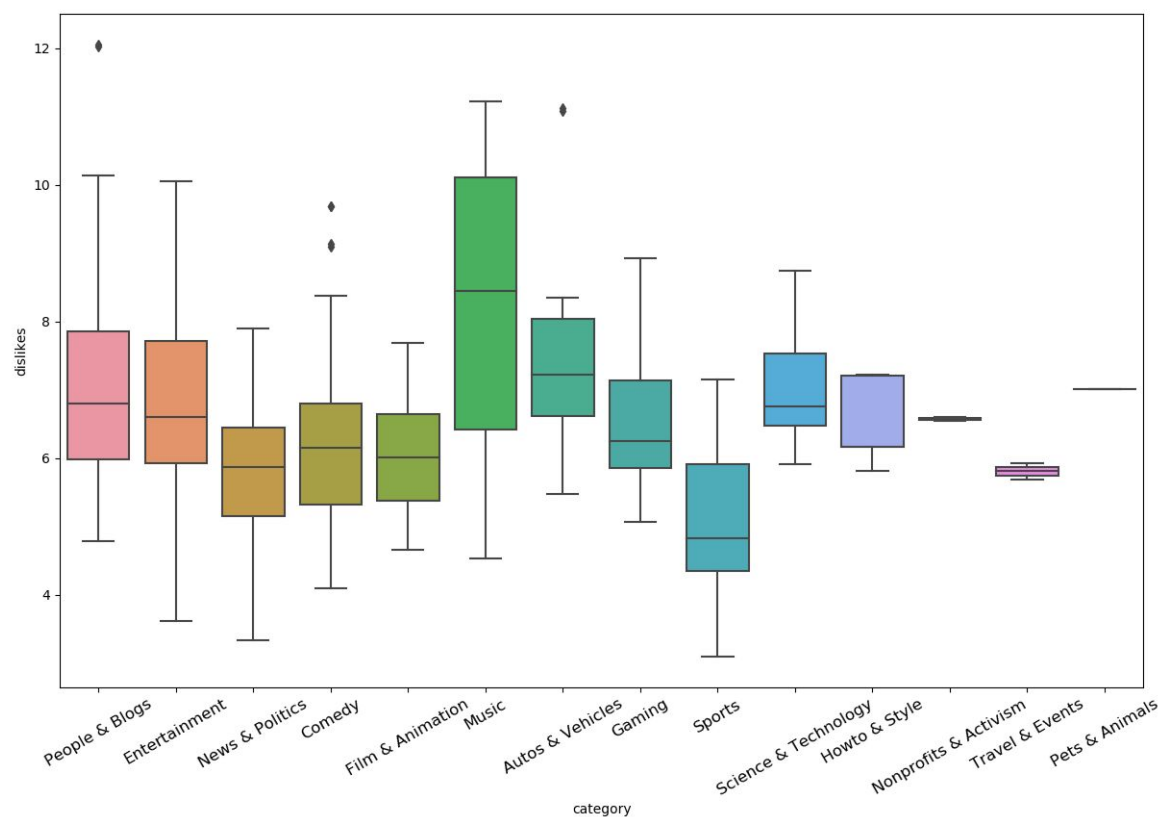
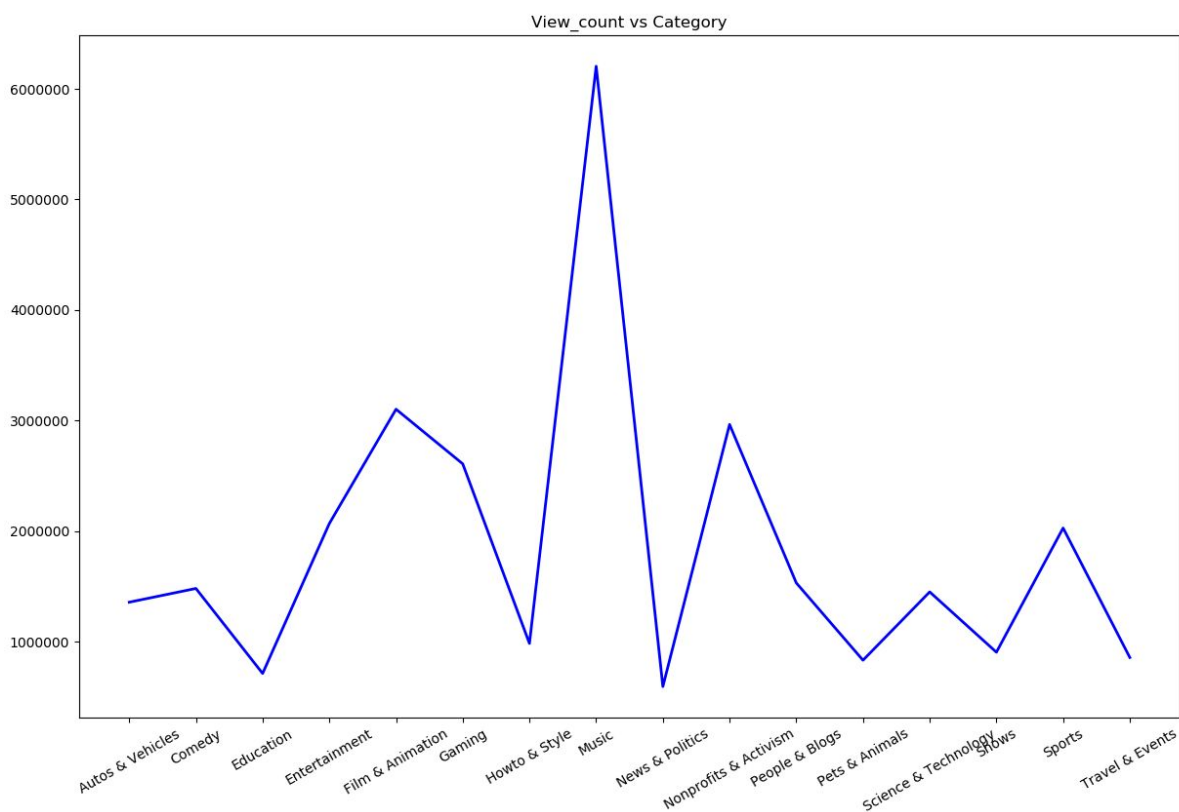
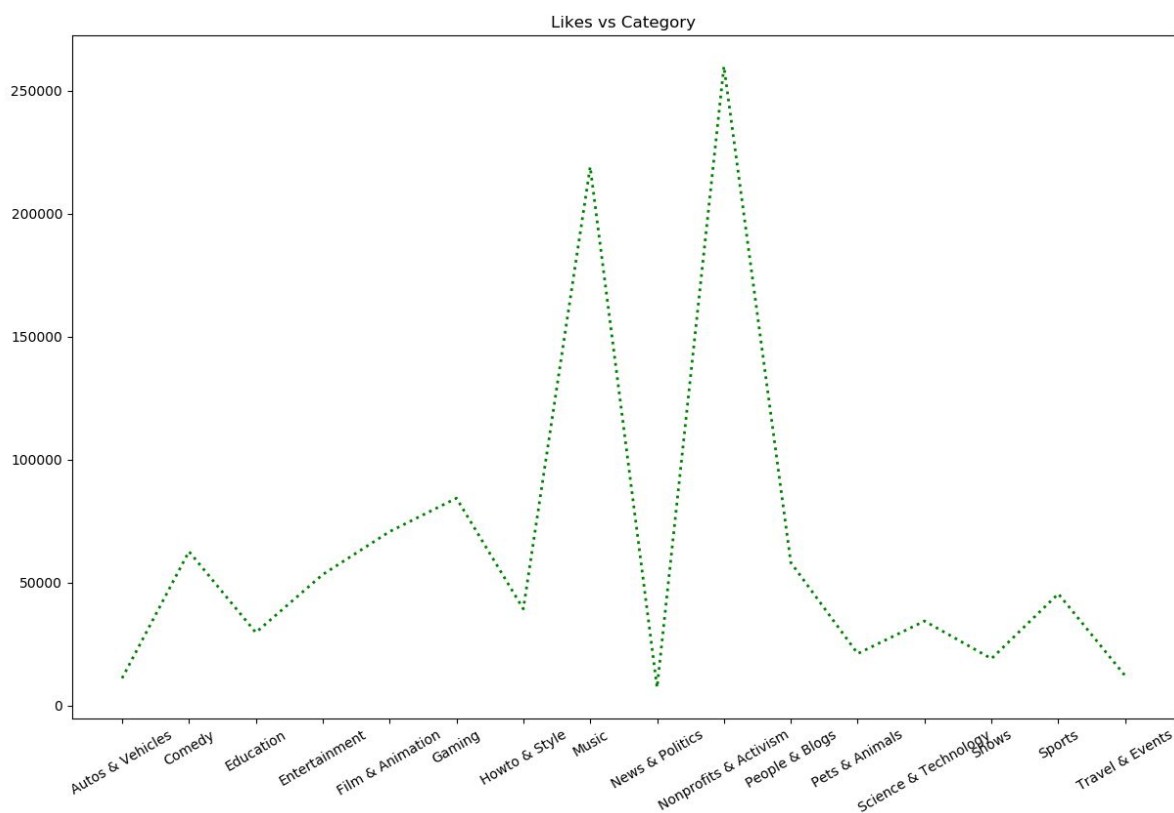


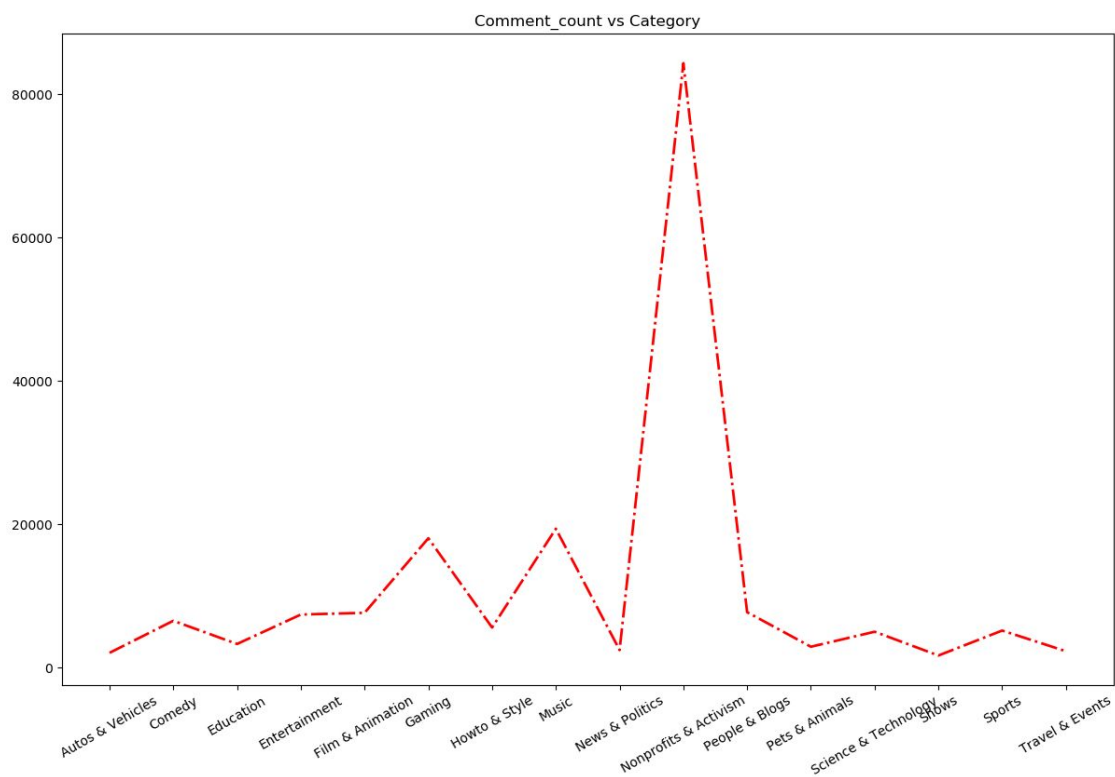
Рис. 9. Діаграма розмаху розподілу неприхильності до відео по категоріям



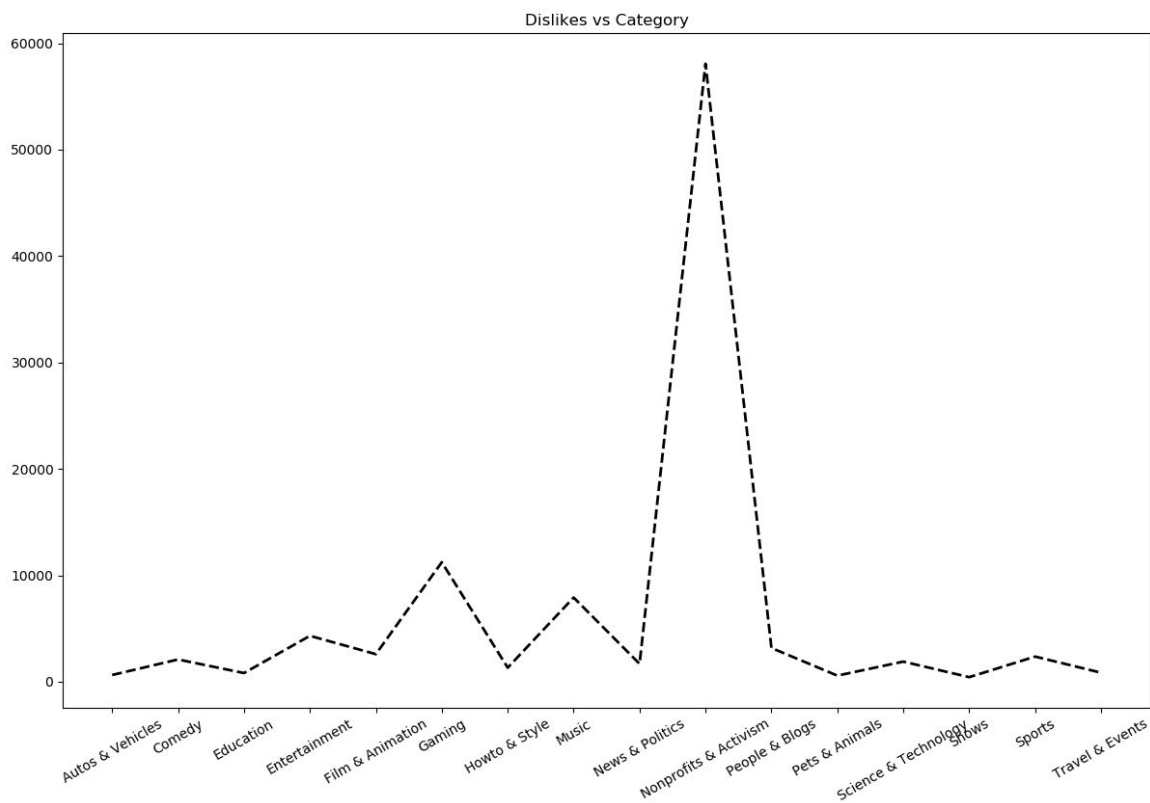
*Рис. 10. Графік середнього рівня переглядів у вигляді ламаної кривої*



*Рис. 11. Графік середнього рівня уподобань у вигляді ламаної кривої*



*Рис. 12. Графік середнього рівня коментарів у вигляді ламаної кривої*



*Рис. 13. Графік середнього рівня неприхильності у вигляді ламаної кривої*

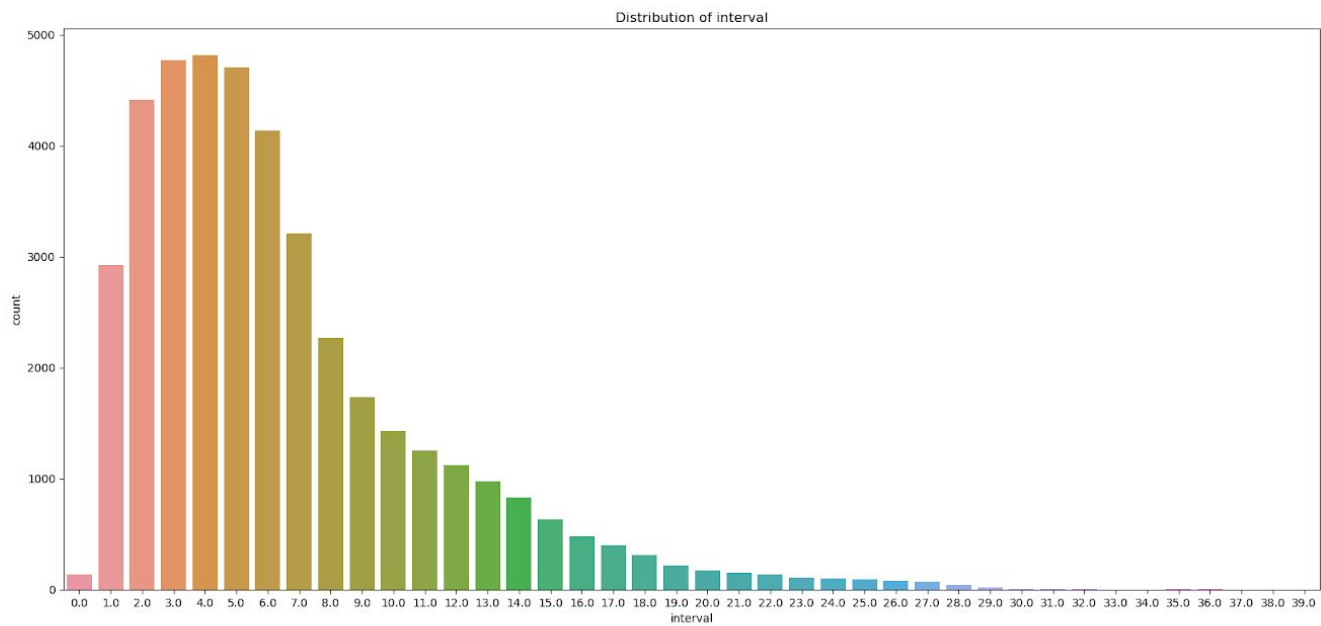


Рис. 14. Гістограма середнього інтервалу в якому відео було в трендах

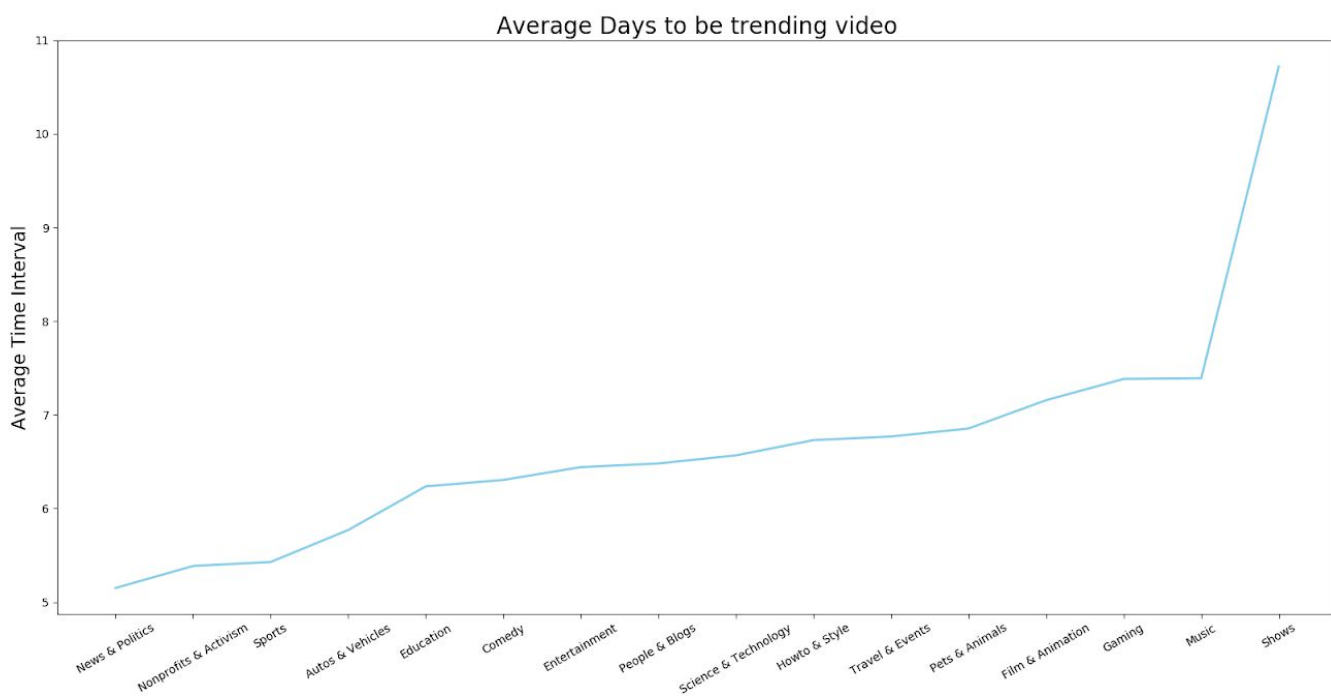


Рис. 15. Графік залежності кількості днів які відео було в трендах від категорії відео

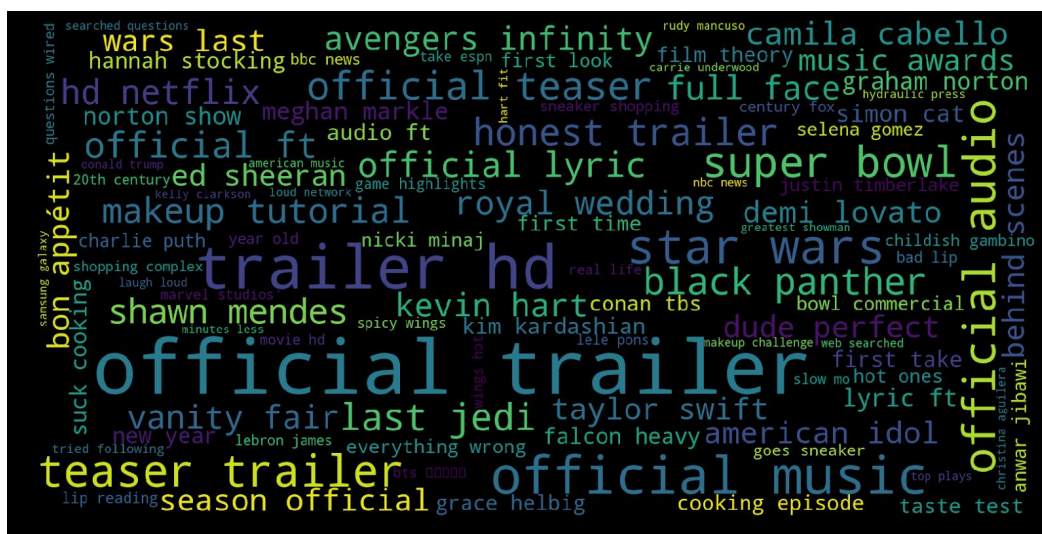


Рис. 16. Хмара тегів на основі описів до відео

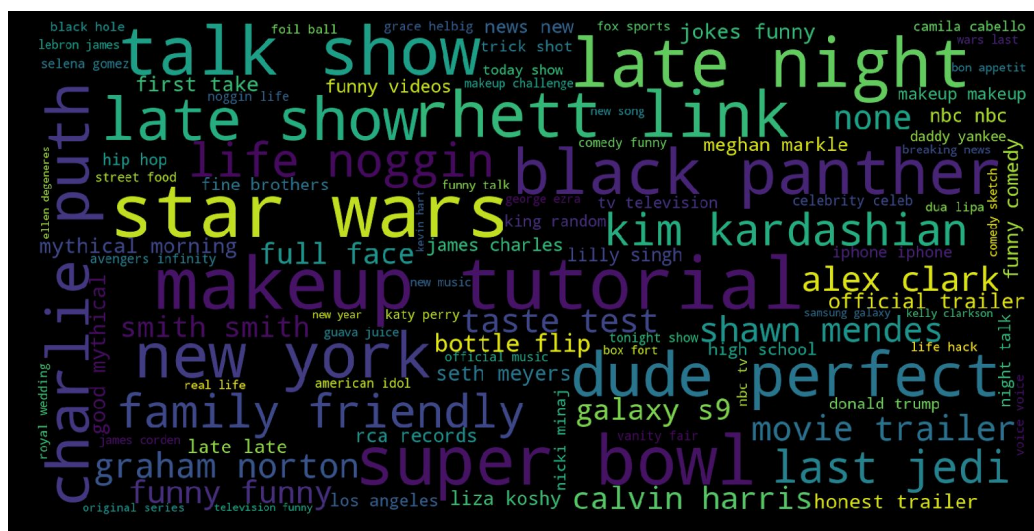
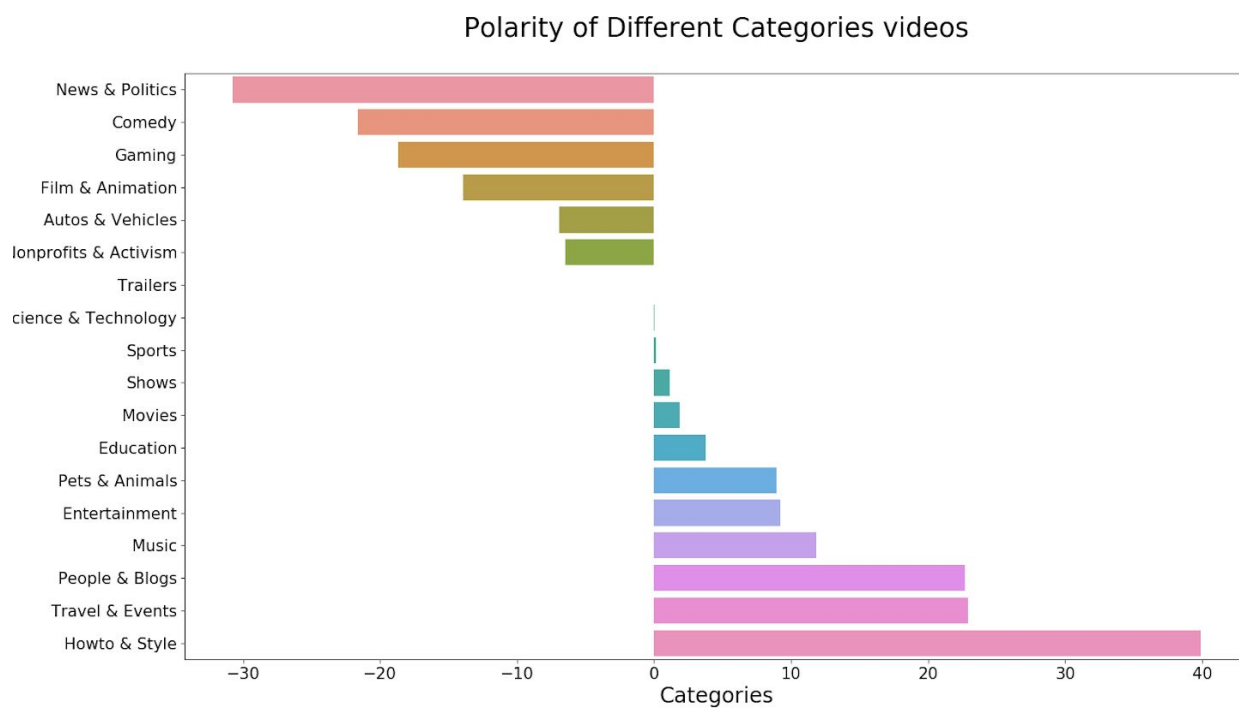


Рис. 17. Хмара тегів на основі тегів до відео



Рис. 18. Хмара тегів на основі назв до відео



*Рис. 19. Графік полярності настроїв відео по категоріям*

## Б. Фрагменти програмного коду

### file\_processing.py

```
def get_videos_data_from_csv(*files) -> list:
    data_frames = []

    for file in files:
        try:
            if file is not None:
                data_frames.append(pd.read_csv(file))
        except FileNotFoundError as e:
            print(e.strerror)

    if len(data_frames) == 0:
        return []

    videos_data = pd.concat(data_frames).to_dict('records')

    return videos_data
```

*Лістинг 1. Алгоритм зчитування даних з датасетів у форматі .csv*

### data\_analysis.py

```
def views_likes_dislikes_comments_normal_distribution(data: DataFrame,
output_dir=Args.analysis_res_dir()):
    data['likes_log'] = np.log(data['likes'] + 1)
    data['view_count_log'] = np.log(data['view_count'] + 1)
    data['dislikes_log'] = np.log(data['dislikes'] + 1)
    data['comment_log'] = np.log(data['comment_count'] + 1)

    plt.figure(figsize=(12, 6))

    plt.subplot(221)
    g1 = sns.distplot(data['view_count_log'])
    g1.set_title("VIEWS LOG DISTRIBUTION", fontsize=16)

    plt.subplot(224)
    g2 = sns.distplot(data['likes_log'], color='green')
    g2.set_title('LIKES LOG DISTRIBUTION', fontsize=16)

    plt.subplot(223)
    g3 = sns.distplot(data['dislikes_log'], color='r')
    g3.set_title("DISLIKES LOG DISTRIBUTION", fontsize=16)

    plt.subplot(222)
    g4 = sns.distplot(data['comment_log'])
    g4.set_title("COMMENTS LOG DISTRIBUTION", fontsize=16)

    plt.subplots_adjust(wspace=0.2, hspace=0.4, top=0.9)

    __save_figure(plt, output_dir, 'normal_distribution.png')
    plt.close()
```

*Лістинг 2. Алгоритм знаходження розподілу кількості переглядів,  
уподобань, неприхильності та коментарів*



### data\_analysis.py

```
def correlation(data: DataFrame, output_dir=Args.analysis_res_dir()):
    corr = data[['view_count', 'likes', 'dislikes', 'comment_count']].corr()
    plot = sns.heatmap(corr, cmap='Blues', annot=True)

    __save_figure(plot.get_figure(), output_dir, 'correlation.png')
    plt.close()
```

*Лістинг 3. Алгоритм знаходження кореляції серед переглядів, уподобань, неприхильності та коментарів*

### data\_analysis.py

```
def category_rating(data: DataFrame, output_dir=Args.analysis_res_dir()):
    plt.figure(figsize=(30, 9))
    plot = sns.countplot(data['category'], order=data['category'].value_counts().index)
    plot.set_title("Counting the Video Category's ", fontsize=20)
    plot.set_xlabel("", fontsize=20)
    plot.set_ylabel("Count", fontsize=20)

    __save_figure(plot.get_figure(), output_dir, 'category_rating.png')
    plt.close()
```

*Лістинг 4. Алгоритм знаходження кількості популярних відео по кожній категорії окремо*

### data\_analysis.py

```
def distribution_boxplot(data: DataFrame, output_dir=Args.analysis_res_dir()):
    view_count = np.log(data['view_count'] + 1)
    likes = np.log(data['likes'] + 1)
    dislikes = np.log(data['dislikes'] + 1)
    comment = np.log(data['comment_count'] + 1)
    data_count = pd.concat([view_count, likes, dislikes, comment], axis=1)
    data_count.index = data['category']
    data_count = data_count[(data_count != 0)]
    plt.figure(figsize=(32, 20))
    plt.subplot(2, 2, 1)
    sns.boxplot(data_count.index, 'view_count', data=data_count,
order=data['category'].value_counts().index)
    plt.xticks(rotation=30, fontsize=12)
    plt.subplot(2, 2, 2)
    sns.boxplot(data_count.index, 'likes', data=data_count,
order=data['category'].value_counts().index)
    plt.xticks(rotation=30, fontsize=12)
    plt.subplot(2, 2, 3)
    sns.boxplot(data_count.index, 'dislikes', data=data_count,
order=data['category'].value_counts().index)
    plt.xticks(rotation=30, fontsize=12)
    plt.subplot(2, 2, 4)
    sns.boxplot(data_count.index, 'comment_count', data=data_count,
order=data['category'].value_counts().index)
    plt.xticks(rotation=30, fontsize=12)
    __save_figure(plt, output_dir, 'distribution_boxplot.png')
    plt.close()
```

*Лістинг 5. Алгоритм знаходження розподілу переглядів, уподобань, неприхильності та коментарів у вигляді діаграми розмаху*

## data\_analysis.py

```
# Plot the distribution of 'view_count', 'likes', 'dislikes', 'comment_count'
def distribution_plot(data: DataFrame, output_dir=Args.analysis_res_dir()):
    general_view = pd.DataFrame(
        data[['view_count', 'likes', 'dislikes',
'comment_count']].groupby(data['category']).mean())
    plt.figure(figsize=(32, 20))
    plt.subplot(2, 2, 1)
    plt.plot(general_view.index, 'view_count', data=general_view, color='blue',
linewidth=2, linestyle='solid')
    plt.title('View_count vs Category')
    plt.xticks(rotation=30)
    plt.subplot(2, 2, 2)
    plt.plot(general_view.index, 'likes', data=general_view, color='green', linewidth=2,
linestyle='dotted')
    plt.title('Likes vs Category')
    plt.xticks(rotation=30)
    plt.subplot(2, 2, 3)
    plt.plot(general_view.index, 'dislikes', data=general_view, color='black',
linewidth=2, linestyle='dashed')
    plt.title('Dislikes vs Category')
    plt.xticks(rotation=30)
    plt.subplot(2, 2, 4)
    plt.plot(general_view.index, 'comment_count', data=general_view, color='red',
linewidth=2, linestyle='dashdot')
    plt.title('Comment_count vs Category')
    plt.xticks(rotation=30)

    __save_figure(plt, output_dir, 'distribution_plot.png')
    plt.close()
```

*Лістинг 6. Алгоритм знаходження розподілу середнього рівня переглядів, уподобань, неприхильності та коментарів у вигляді ламаної кривої*

## data\_analysis.py

```
# The distribution of days that videos take to become popular
def distribution_of_days_preprocessing(data: DataFrame):
    data['published_at'] = pd.to_datetime(data['published_at'], errors='coerce',
format='%Y-%m-%dT%H:%M:%S.%fZ')
    data['trending_date'] = pd.to_datetime(data['trending_date'], errors='coerce',
format='%Y.%d.%m')

    data['publish_date'] = data['published_at'].dt.date
    data['publish_time'] = data['published_at'].dt.time
    data['interval'] = (data['trending_date'].dt.date -
data['publish_date']).astype('timedelta64[D]')
    data['interval'] = data[data['interval'] < 40]['interval']

    return data

# Histogram of distribution of interval
def distribution_of_days_histogram(data: DataFrame, output_dir=Args.analysis_res_dir()):
    data = distribution_of_days_preprocessing(data)

    plt.figure(figsize=(25, 9))

    plot = sns.countplot(data['interval'])
    plt.title('Distribution of interval')

    __save_figure(plot.get_figure(), output_dir, 'distribution_of_days_histogram.png')
    plt.close()

# Average time interval between published and trending
def distribution_of_average_time(data: DataFrame, output_dir=Args.analysis_res_dir()):
    data = distribution_of_days_preprocessing(data)
```

```

df_t =
pd.DataFrame(data['interval'].groupby(data['category']).mean()).sort_values(by="interval"
)
plt.figure(figsize=(25, 9))
plt.plot(df_t, color='skyblue', linewidth=2)
plt.title("Average Days to be trending video", fontsize=20)
plt.xlabel('Category', fontsize=16)
plt.ylabel('Average Time Interval', fontsize=16)
plt.xticks(rotation=30)

__save_figure(plt, output_dir, 'distribution_of_average_time.png')
plt.close()

```

*Лістинг 7. Алгоритм знаходження середнього інтервалу в днях в якому відео перебувало в трендах у вигляді гістограми та графіка залежності середньої кількості днів від жанру відео*

#### data\_analysis.py

```

# Top videos whose view_count grow fastest among categories
def view_count_fastest_grow_among_categories(data: DataFrame, preprocessing=True):
    # calculate growth rate for each video
    data['growth_rate_view'] = data['view_count'] / (data['interval'] + 1)

    df = data.set_index(keys='title').groupby(by=['category'])['channel_title',
'growth_rate_view'].apply(
        lambda g: g.nlargest(5, 'growth_rate_view'))
    return df.head(90)

# Top videos whose likes grow fastest among categories
def likes_fastest_grow_among_categories(data: DataFrame, preprocessing=True):
    # calculate growth rate for each video
    data['growth_rate_like'] = data['likes'] / (data['interval'] + 1)

    df = data.set_index(keys='title').groupby(by=['category'])['channel_title',
'growth_rate_like'].apply(
        lambda g: g.nlargest(5, 'growth_rate_like'))
    return df.head(90)

# Top videos whose dislikes grow fastest among categories
def dislikes_fastest_grow_among_categories(data: DataFrame, preprocessing=True):
    # calculate growth rate for each video
    data['growth_rate_dislike'] = data['dislikes'] / (data['interval'] + 1)

    df = data.set_index(keys='title').groupby(by=['category'])['channel_title',
'growth_rate_dislike'].apply(
        lambda g: g.nlargest(5, 'growth_rate_dislike'))
    return df.head(90)

# Top videos whose comment_count grow fastest among categories
def comment_count_fastest_grow_among_categories(data: DataFrame, preprocessing=True):
    # calculate growth rate for each video
    data['growth_rate_comment'] = data['comment_count'] / (data['interval'] + 1)

    df = data.set_index(keys='title').groupby(by=['category'])['channel_title',
'growth_rate_comment'].apply(
        lambda g: g.nlargest(5, 'growth_rate_comment'))
    return df.head(90)

```

*Лістинг 8. Алгоритми знаходження каналів по категоріям з найбільшою швидкістю зростання переглядів, уподобань, неприхильності та коментарів*

## data\_analysis.py

```
def top_channels(data: DataFrame, num_of_channels: int):
    channel = pd.DataFrame(data['channel_title'].groupby(by=[data['channel_title'],
data['category']])).count())
    channel.columns = ['count']
    top_channel = channel.nlargest(num_of_channels, ['count'])
    return top_channel.head(num_of_channels)
```

*Лістинг 9. Алгоритми знаходження каналів чиї відео найчастіше  
потрапляють в тренди*

## data\_analysis.py

```
def word_cloud_for_tags(data: DataFrame, output_dir=Args.analysis_res_dir()):
    tags_word = data['tags'].str.lower().str.cat(sep=' ')

    __create_and_save_word_cloud(
        data=tags_word,
        filename='word_cloud_for_tags.png',
        user_stopwords=USER_STOPWORDS,
        output_dir=output_dir
    )

    del tags_word

def word_cloud_for_titles(data: DataFrame, output_dir=Args.analysis_res_dir()):
    title_word = data['title'].str.lower().str.cat(sep=' ')

    __create_and_save_word_cloud(
        data=title_word,
        filename='word_cloud_for_titles.png',
        user_stopwords=USER_STOPWORDS,
        output_dir=output_dir
    )

    del title_word

def word_cloud_for_description(data: DataFrame, output_dir=Args.analysis_res_dir()):
    description_word = data['title'].str.lower().str.cat(sep=' ')

    __create_and_save_word_cloud(
        data=description_word,
        filename='word_cloud_for_description.png',
        user_stopwords=USER_STOPWORDS,
        output_dir=output_dir
    )

    del description_word
```

*Лістинг 10. Алгоритм побудови хмари тегів для назв, опису та тегів відео*

## data\_analysis.py

```
def sentiment_analysis(data: DataFrame, output_dir=Args.analysis_res_dir()):
    category_list = data['category'].unique()

    # Collect all the related stopwords.
    en_stopwords = list(stopwords.words('english'))
    de_stopwords = list(stopwords.words('german'))
    fr_stopwords = list(stopwords.words('french'))
    ru_stopwords = list(stopwords.words('russian'))

    en_stopwords.extend(de_stopwords)
    en_stopwords.extend(fr_stopwords)
    en_stopwords.extend(ru_stopwords)

    polarities = list()
    MAX_N = 10000

    for i in category_list:
        print(f'>> {i}')

        tags_word = data[data['category'] == i]['tags'].str.lower().str.cat(sep=' ')

        # removes punctuation,numbers and returns list of words
        tags_word = re.sub('[^A-Za-z]+', ' ', tags_word)
        word_tokens = word_tokenize(tags_word)
        filtered_sentence = [w for w in word_tokens if not w in en_stopwords]
        without_single_chr = [word for word in filtered_sentence if len(word) > 2]

        # Remove numbers
        cleaned_data_title = [word for word in without_single_chr if not word.isdigit()]

        # Calculate frequency distribution
        word_dist = nltk.FreqDist(cleaned_data_title)
        hnhk = pd.DataFrame(word_dist.most_common(MAX_N),
                            columns=['Word', 'Frequency'])

        compound = .0
        for word in hnhk['Word'].head(MAX_N):
            compound += SentimentIntensityAnalyzer().polarity_scores(word)['compound']

        polarities.append(compound)

    category_list = pd.DataFrame(category_list)
    polarities = pd.DataFrame(polarities)
    tags_sentiment = pd.concat([category_list, polarities], axis=1)
    tags_sentiment.columns = ['category', 'polarity']
    tags_sentiment = tags_sentiment.sort_values('polarity').reset_index()

    plt.figure(figsize=(18, 10))
    sns.barplot(x=tags_sentiment['polarity'], y=tags_sentiment['category'],
                data=tags_sentiment)

    plt.xlabel("Categories", fontsize=20)
    plt.ylabel("Polarity", fontsize=20)
    plt.yticks(fontsize=15)
    plt.xticks(fontsize=15)
    plt.title("\nPolarity of Different Categories videos\n", fontsize=25)

    __save_figure(plt, output_dir, 'polarity_of_categories.png')
    plt.close()
```

*Лістинг 11. Алгоритм аналізу полярності настроїв по категоріям*