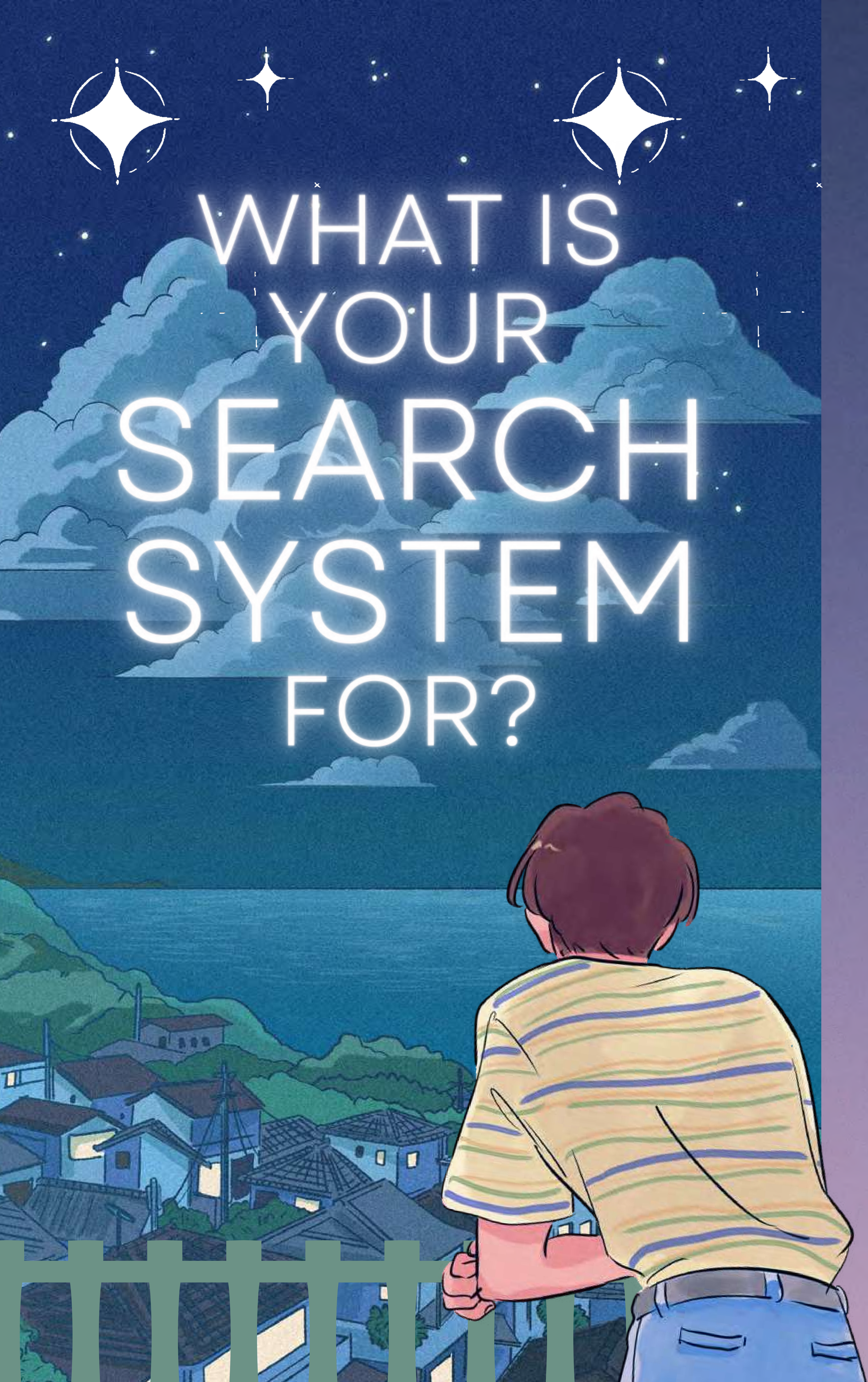
ANIME

SEARCH ENGINE

Presented by:

- Ramita Deeprom 6488018
- Thitiwut Harnphatcharapanukorn 6488025
- Burit Sihabut 6488079
- Pongsakorn Kongkaewrasamee 6488215



WHAT IS YOUR SEARCH SYSTEM FOR?

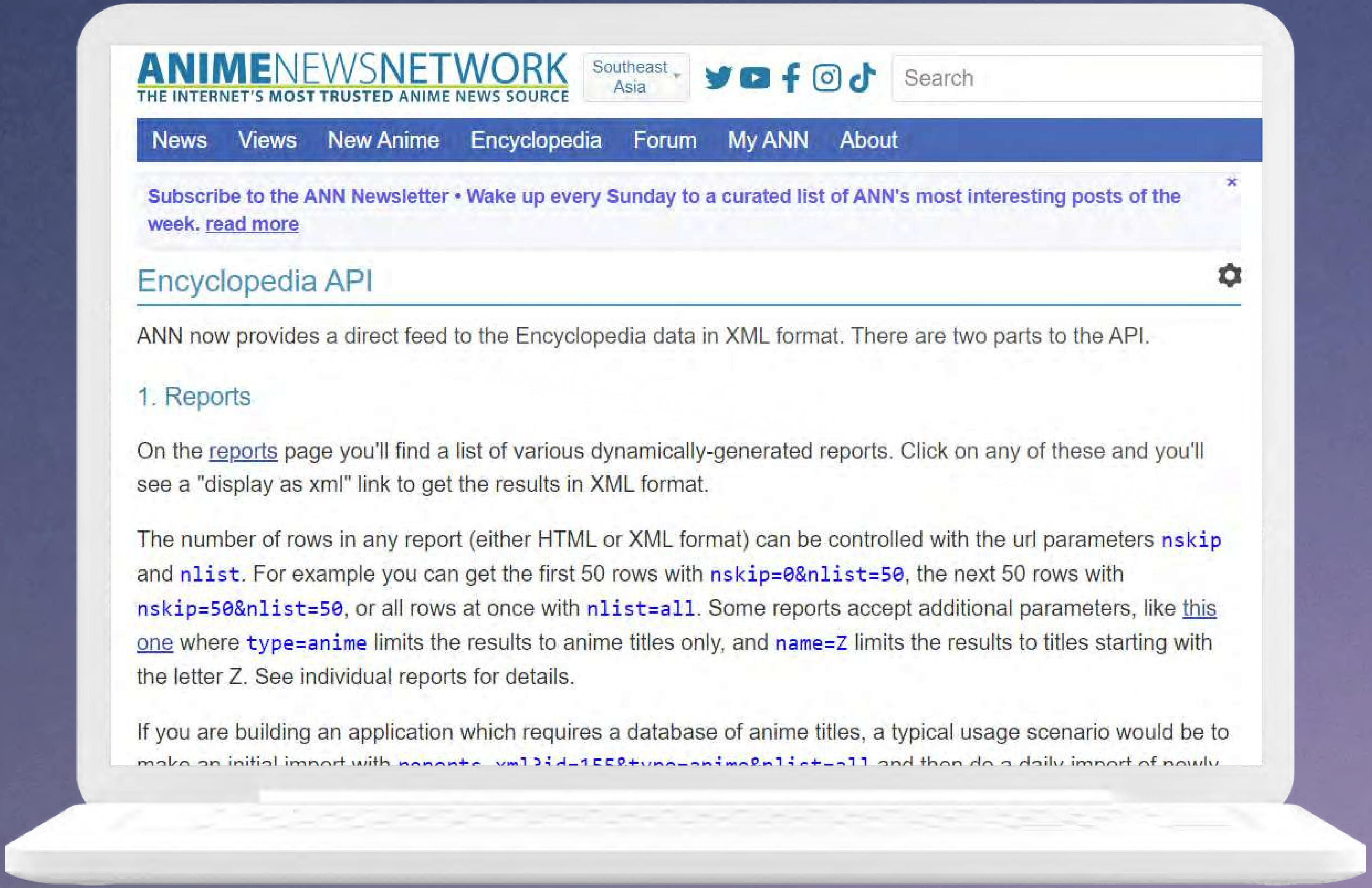
SEARCH PURPOSES

- Explore Anime from a vast amount of data
- Retrieve detailed information about anime series and movies

TARGET USERS

- ANIME LOVERS
- CASUAL VIEWERS
- ANIME INDUSTRY PROFESSIONALS

DATA SOURCE



Subscribe to the ANN Newsletter • Wake up every Sunday to a curated list of ANN's most interesting posts of the week. read more

Encyclopedia API

ANN now provides a direct feed to the Encyclopedia data in XML format. There are two parts to the API.

1. Reports

On the reports page you'll find a list of various dynamically-generated reports. Click on any of these and you'll see a "display as xml" link to get the results in XML format.

The number of rows in any report (either HTML or XML format) can be controlled with the url parameters `nskip` and `nlist`. For example you can get the first 50 rows with `nskip=0&nlist=50`, the next 50 rows with `nskip=50&nlist=50`, or all rows at once with `nlist=all`. Some reports accept additional parameters, like this one where `type=anime` limits the results to anime titles only, and `name=Z` limits the results to titles starting with the letter Z. See individual reports for details.

If you are building an application which requires a database of anime titles, a typical usage scenario would be to make an initial import with `reports.xml?id=155&type=anime&nlist=all` and then do a daily import of newly

ANIMENEWSNETWORK
THE INTERNET'S MOST TRUSTED ANIME NEWS SOURCE

Anime News Network API provides a free API that returns an XML result.

XML

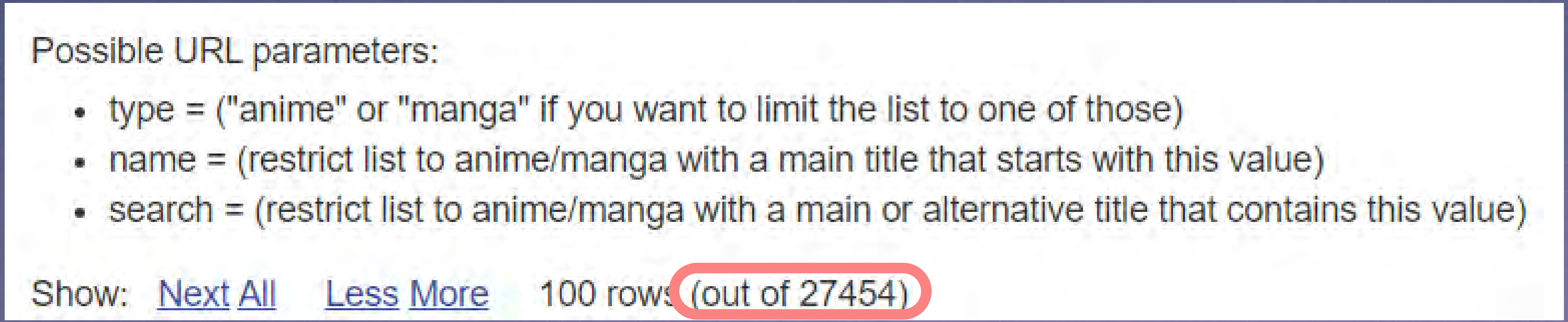
https://www.animenewsnetwork.com/encyclopedia/api.php

DATA SOURCE

For example, this is a URL to fetch for information of an anime

Document ID

https://cdn.animenewsnetwork.com/encyclopedia/api.xml?anime=454



Possible URL parameters:

- type = ("anime" or "manga" if you want to limit the list to one of those)
- name = (restrict list to anime/manga with a main title that starts with this value)
- search = (restrict list to anime/manga with a main or alternative title that contains this value)

Show: Next All Less More 100 rows (out of 27454)

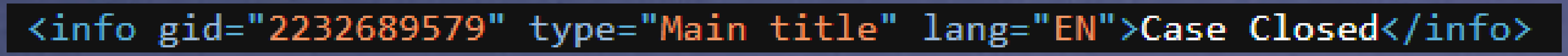
There are over 20,000 documents for us to do this project.

Let's say we will just random around 250+- anime IDs for this project

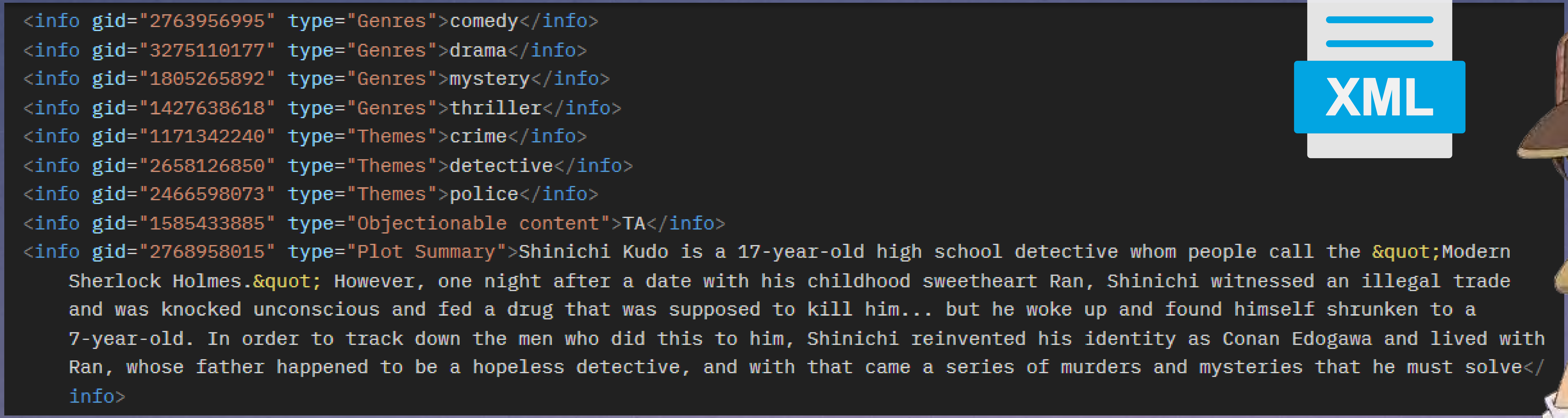
DATA SOURCE

Output

https://cdn.animenewsnetwork.com/encyclopedia/api.xml?anime=**454**



```
<info gid="2232689579" type="Main title" lang="EN">Case Closed</info>
```

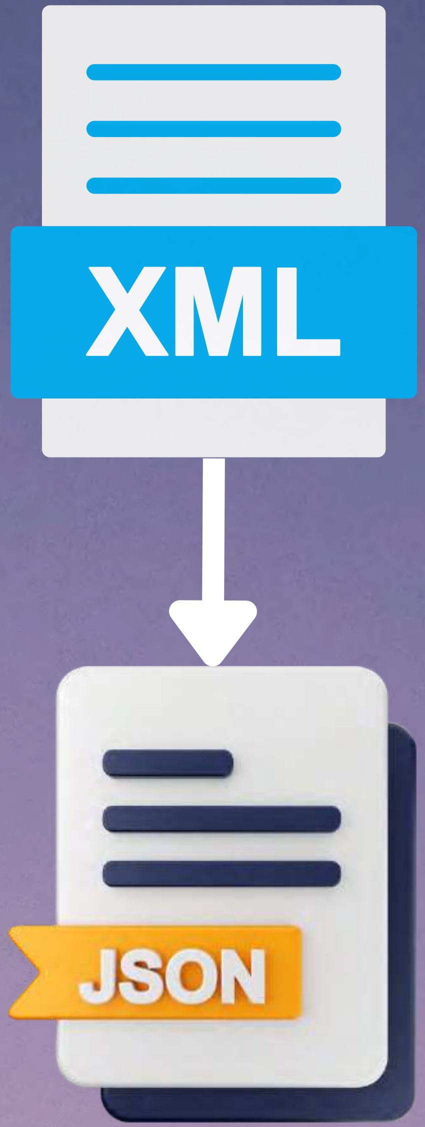


```
<info gid="2763956995" type="Genres">comedy</info>
<info gid="3275110177" type="Genres">drama</info>
<info gid="1805265892" type="Genres">mystery</info>
<info gid="1427638618" type="Genres">thriller</info>
<info gid="1171342240" type="Themes">crime</info>
<info gid="2658126850" type="Themes">detective</info>
<info gid="2466598073" type="Themes">police</info>
<info gid="1585433885" type="Objectionable content">TA</info>
<info gid="2768958015" type="Plot Summary">Shinichi Kudo is a 17-year-old high school detective whom people call the &quot;Modern
    Sherlock Holmes.&quot; However, one night after a date with his childhood sweetheart Ran, Shinichi witnessed an illegal trade
    and was knocked unconscious and fed a drug that was supposed to kill him... but he woke up and found himself shrunken to a
    7-year-old. In order to track down the men who did this to him, Shinichi reinvented his identity as Conan Edogawa and lived with
    Ran, whose father happened to be a hopeless detective, and with that came a series of murders and mysteries that he must solve</
    info>
```

We will index the document from "Main titile", "Plot Summary", "Genres", and "Themes"

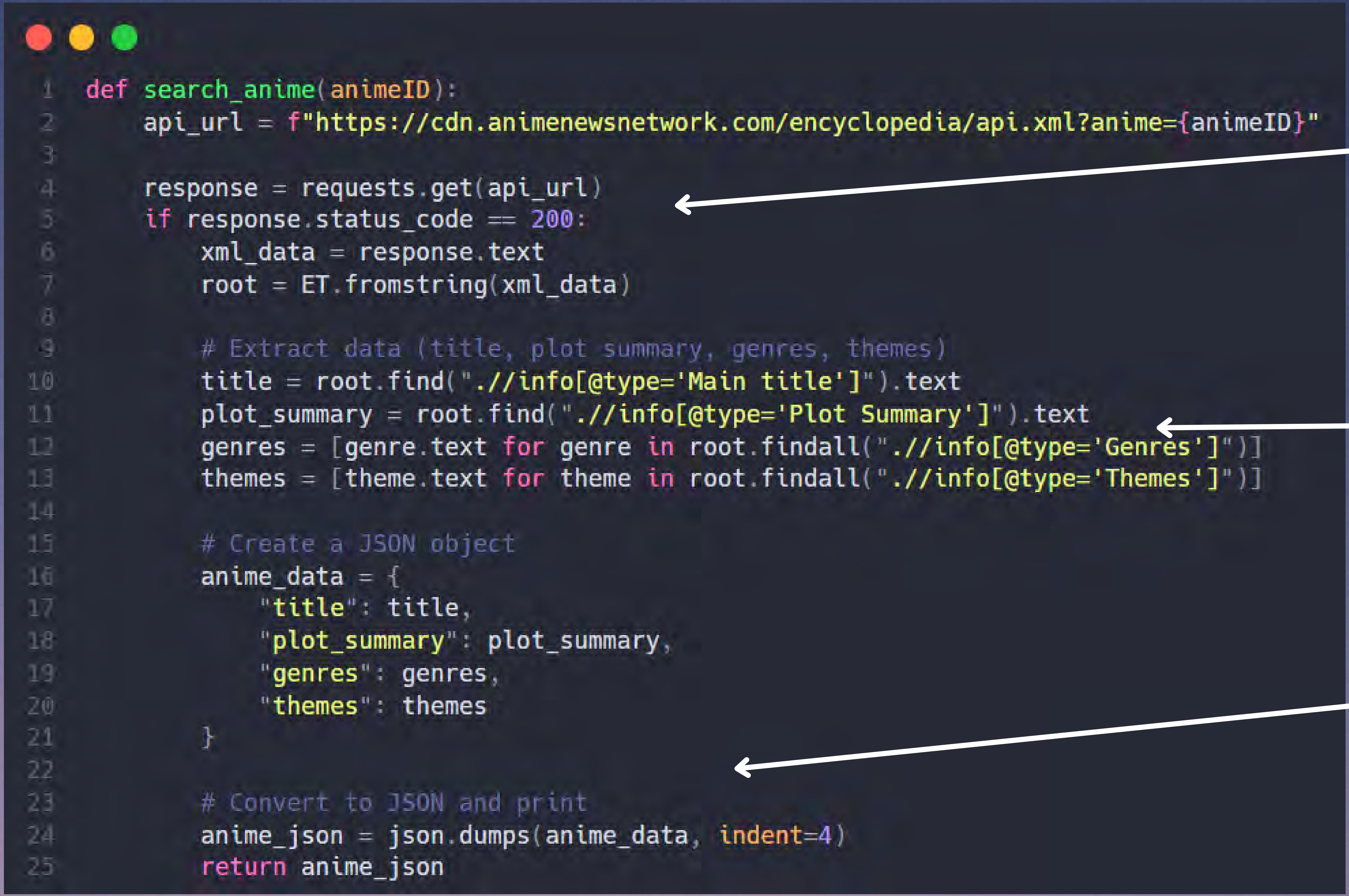
DATA PREPROCESSING

FOR EACH DOCUMENT



We will extract only the essential part from the XML document and convert them into JSON.
This extracted document will later be used in indexing

DATA PREPROCESSING



```python
def search_anime(animeID):
    api_url = f"https://cdn.animenewsnetwork.com/encyclopedia/api.xml?anime={animeID}"

    response = requests.get(api_url)
    if response.status_code == 200:
        xml_data = response.text
        root = ET.fromstring(xml_data)

        # Extract data (title, plot summary, genres, themes)
        title = root.find(".//info[@type='Main title']").text
        plot_summary = root.find(".//info[@type='Plot Summary']").text
        genres = [genre.text for genre in root.findall(".//info[@type='Genres']")]
        themes = [theme.text for theme in root.findall(".//info[@type='Themes']")]

        # Create a JSON object
        anime_data = {
            "title": title,
            "plot_summary": plot_summary,
            "genres": genres,
            "themes": themes
        }

        # Convert to JSON and print
        anime_json = json.dumps(anime_data, indent=4)
        return anime_json
```

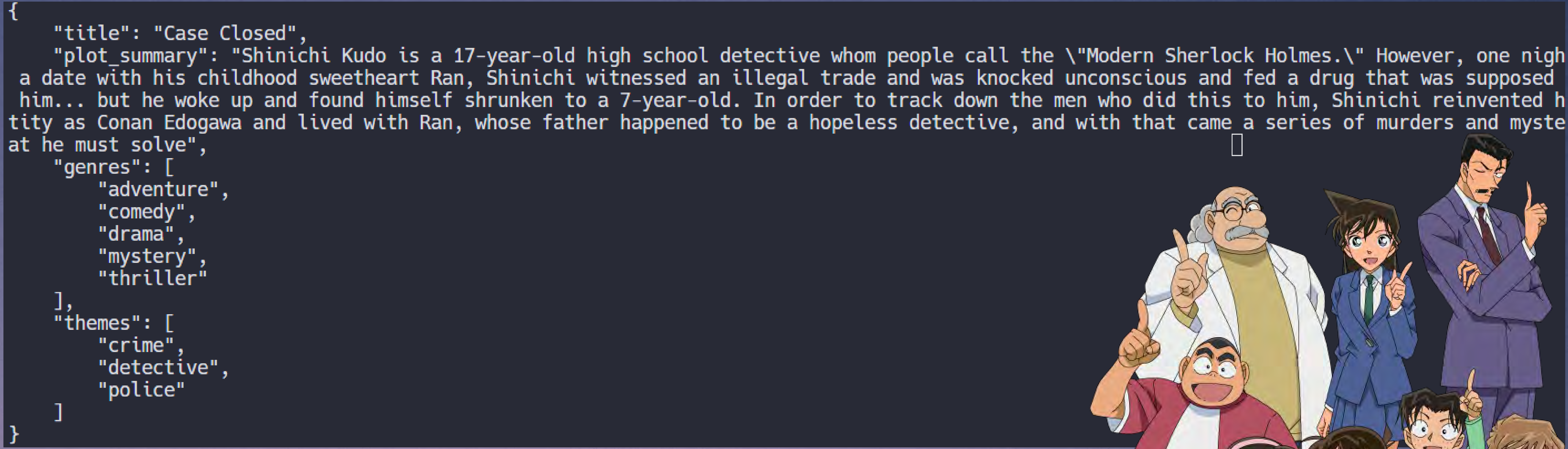
URL Request

Extract necessary parts from the XML result

Convert into JSON

DATA PREPROCESSING

Example JSON Output

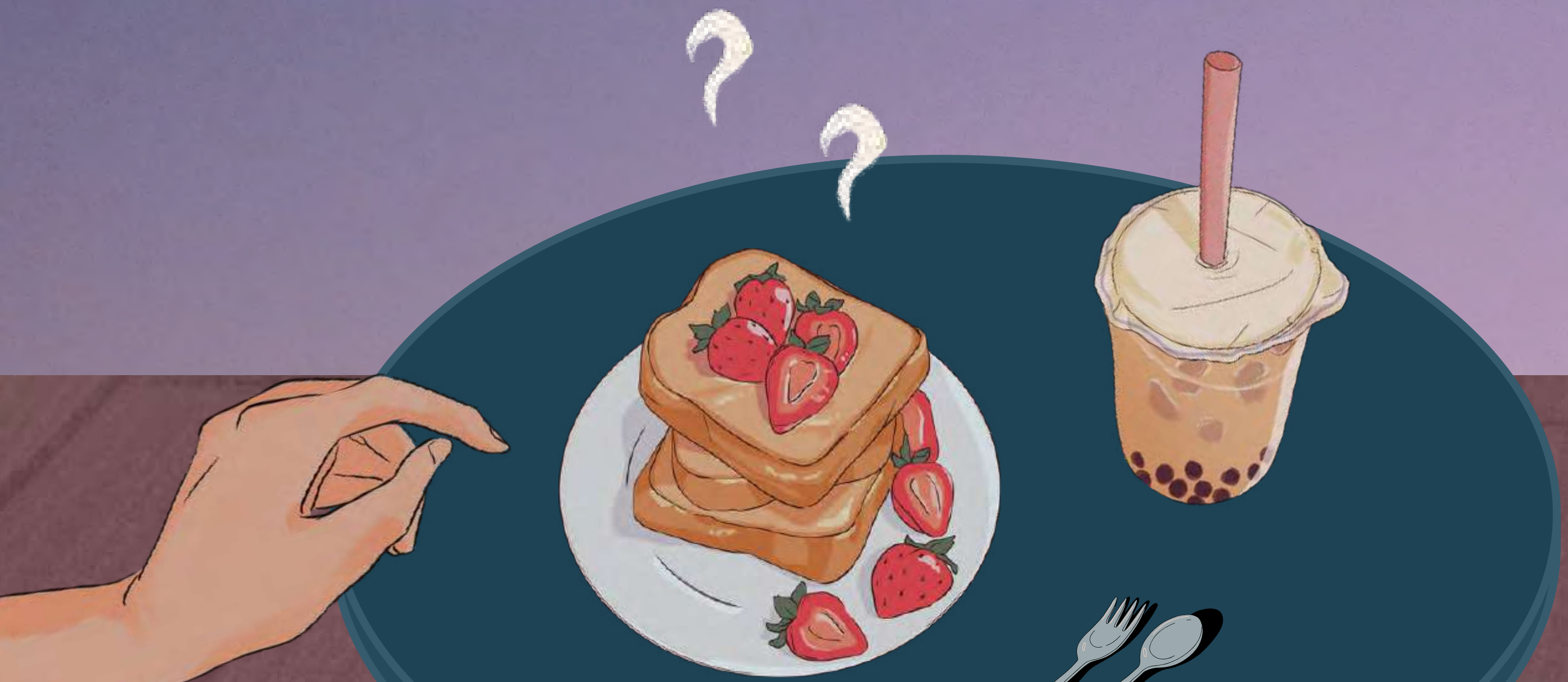


```
{
    "title": "Case Closed",
    "plot_summary": "Shinichi Kudo is a 17-year-old high school detective whom people call the \"Modern Sherlock Holmes.\" However, one nigh
 a date with his childhood sweetheart Ran, Shinichi witnessed an illegal trade and was knocked unconscious and fed a drug that was supposed
 him... but he woke up and found himself shrunken to a 7-year-old. In order to track down the men who did this to him, Shinichi reinvented h
tity as Conan Edogawa and lived with Ran, whose father happened to be a hopeless detective, and with that came a series of murders and myste
at he must solve",
    "genres": [
        "adventure",
        "comedy",
        "drama",
        "mystery",
        "thriller"
    ],
    "themes": [
        "crime",
        "detective",
        "police"
    ]
}
```

DOCUMENT INDEXING

elasticsearch

We will use elasticsearch to create an Inverted Index for our documents



DOCUMENT RANKING

We will also use elasticsearch for ranking the documents.

elasticsearch

From our research, elasticsearch uses TF-IDF and BM25 ranking algorithm

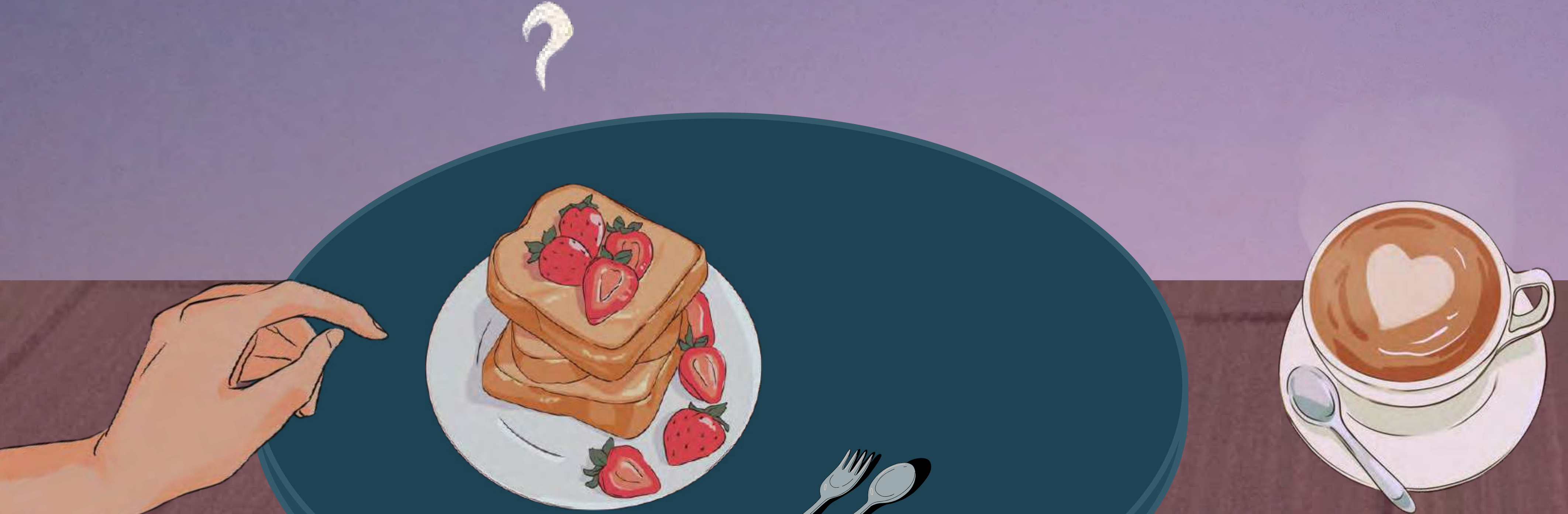
If possible, we want to also weight each field of the document.

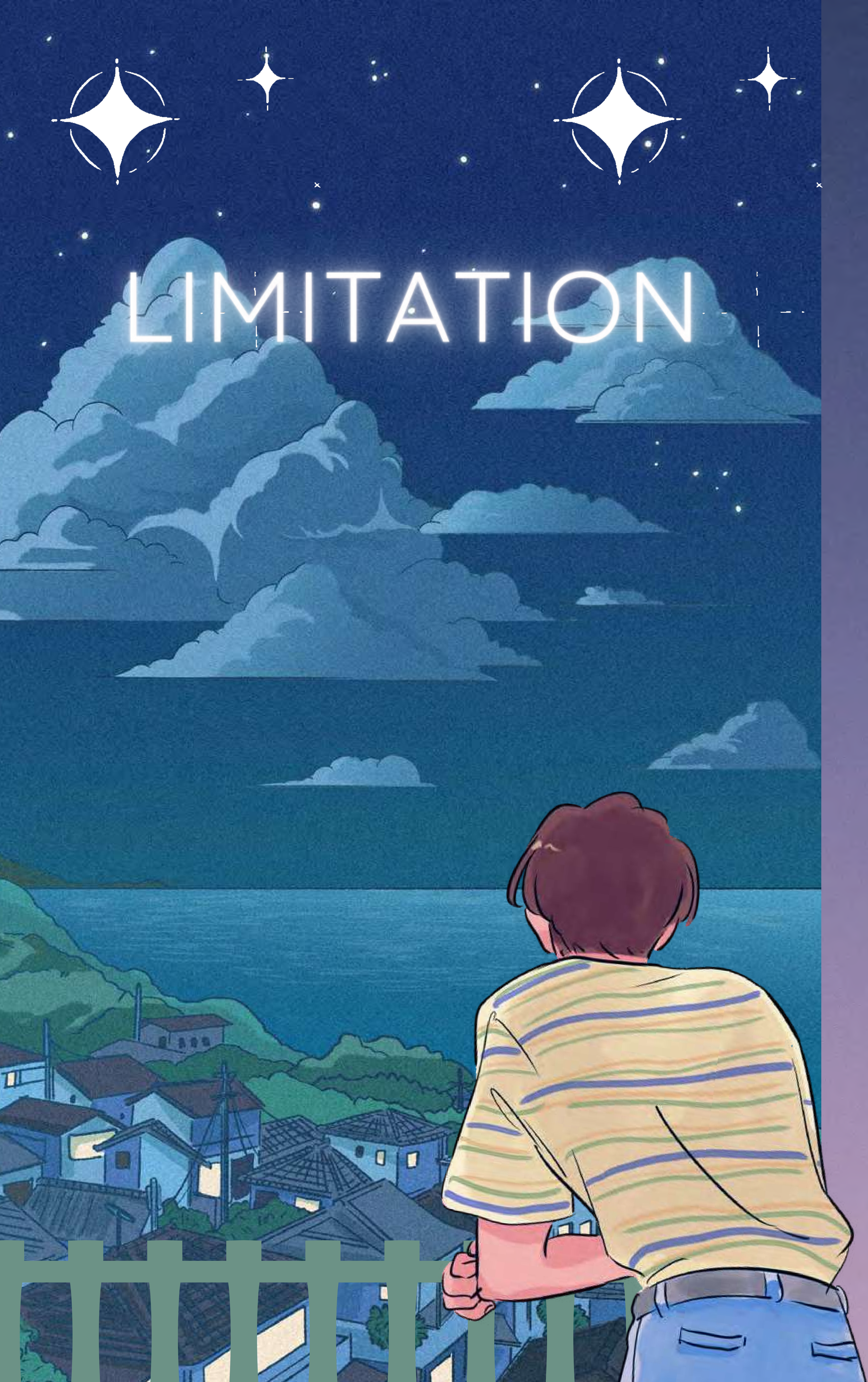
Title > Summary > Genres = Themes

THANK YOU FOR LISTENING!

Don't hesitate to ask any questions!

DEMONSTRATION





LIMITATION

Elasticsearch

- DATA CONSISTENCY
- SCALABILITY

Git and GitHub

- VERSION CONTROL

DIFFICULTIES AND CHALLENGES

- QUERY OPTIMIZATION
- DIFFICULTIES ACCESSING KIBANA
- LONG DATA COLLECTION PERIOD
- HTML/XML TAGS NOT FILTERED
- GIT WORKFLOW
- MONITORING AND MAINTENANCE

LESSONS LEARNED

elasticsearch

kibana

Git + GitHub

THANK YOU FOR LISTENING!

Don't hesitate to ask any questions!