

# BÁO CÁO CUỐI KỲ



## HỆ THỐNG ĐỀ XUẤT ANIME

# Nhóm 1

## THÀNH VIÊN NHÓM



MSSV	Họ và tên
221124029233	Phạm Huỳnh Bảo Phương
221124029227	Trần Thị Minh Ngọc
221124029253	Trần Thị Nhuý
221124029252	Trần Thị Thảo Vy

# BÁO CÁO CUỐI KỲ

# NỘI DUNG



## R1: CÀO DỮ LIỆU

Thực hiện bằng Selenium và Request

2

## R2: ĐƯA DỮ LIỆU VÀO CƠ SỞ DỮ LIỆU

Thực hiện kết nối với cơ sở dữ liệu bằng Python và nối bảng

4

## R3: TIỀN XỬ LÝ VÀ PHÂN CHIA CẤU TRÚC

Sử dụng các hàm và thủ tục

Tách bảng bằng các mối quan hệ của nó

5

## R4: SAO LƯU DỮ LIỆU

Thực hiện sao lưu cơ sở dữ liệu bằng T-sql và thao tác

8

## R5: PHÂN QUYỀN CHO HỆ QUẢN TRỊ

Thực hiện phân quyền cho cơ sở dữ liệu bằng T-sql và thao tác

10

## R6: TRỰC QUAN HOÁ DỮ LIỆU

Sử dụng Tableau, Power BI và Python

14

## R7: XÂY DỰNG HỆ THỐNG ĐỀ XUẤT ANIME

Xây dựng hệ thống bằng Python và đưa lên website

15



# LÍ DO CHỌN ĐỀ TÀI

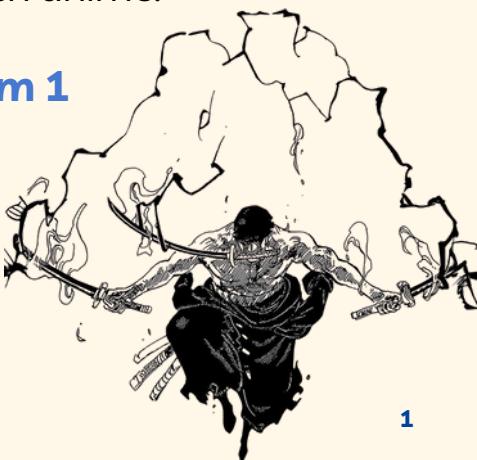


Ngành công nghiệp anime hiện nay đang phát triển mạnh mẽ với số lượng lớn các tác phẩm ra đời mỗi năm, mang lại sự đa dạng về thể loại và nội dung. Tuy nhiên, điều này cũng khiến người xem gặp khó khăn trong việc tìm kiếm những bộ anime phù hợp với sở thích của mình. Đề tài "**Xây dựng hệ thống đề xuất anime**" được lựa chọn nhằm giải quyết vấn đề này, giúp người dùng dễ dàng khám phá những tác phẩm phù hợp và tối ưu hóa trải nghiệm giải trí cá nhân.

Hệ thống đề xuất không chỉ hỗ trợ người dùng tìm kiếm nội dung một cách nhanh chóng, hiệu quả mà còn góp phần nâng cao giá trị của các nền tảng cung cấp anime, thông qua việc cải thiện sự hài lòng và mức độ tương tác của người xem. Đây là một giải pháp thiết thực, đáp ứng nhu cầu ngày càng cao về cá nhân hóa trong lĩnh vực giải trí.

Với mục tiêu kết nối người xem với những bộ anime phù hợp nhất, đề tài này không chỉ mang lại lợi ích thực tế mà còn thể hiện sự kết hợp giữa niềm đam mê với anime và mong muốn tạo ra những giá trị ý nghĩa cho cộng đồng yêu thích anime.

## Nhóm 1



## WEBSITE ĐỂ LẤY DỮ LIỆU VÀ CÁC TRƯỜNG DỮ LIỆU CẦN LẤY

Website mà nhóm lựa chọn: <https://myanimelist.net/>

Rank	Title	Score	Your Score	Status
1	<b>Sousou no Frieren</b> (TV (28 eps)) Sep 2023 - Mar 2024 963,148 members	★ 9.32	N/A	Add to My List
2	<b>Fullmetal Alchemist: Brotherhood</b> (TV (64 eps)) Apr 2009 - Jul 2010 3,447,406 members	★ 9.10	N/A	Add to My List

Các trường nhóm muốn sử dụng

## SCRAPING DỮ LIỆU BẰNG PYTHON

### REQUEST

Requests là một mô-đun Python mà bạn có thể sử dụng để gửi tất cả các loại yêu cầu HTTP. Đây là một thư viện dễ sử dụng với nhiều tính năng khác nhau, từ việc truyền các tham số trong URL cho đến gửi các header tùy biến và Xác minh SSL.

### SELENIUM

Selenium có thể tự động hóa trình duyệt và tương tác với trang web trong thời gian thực từ đó giúp thu thập dữ liệu từ các trang web động mà không cần phải tải trước nội dung HTML.



## REQUEST

- Import các thư viện cần thiết:

```
import requests
from bs4 import BeautifulSoup
from sqlalchemy import create_engine, Table, Column, Integer, Float, String, MetaData, insert
import urllib
import time
import re
```

- Giả lập trình duyệt

```
header = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36 coc_coc_browser/132.0.0'}
```

- Kết nối SQL khởi tạo metadata và cung cấp URL của trang web cào

```
# URL trang web cần scrape
base_url = "https://myanimelist.net/topanime.php"
```

- Hàm thử lại nếu kết nối đến trang web lỗi

```
# Hàm retry để gửi request với số lần thử lại
def fetch_url(url, max_retries=3, sleep_time=5):
    for attempt in range(max_retries):
        try:
            response = requests.get(url, headers=header)
            if response.status_code == 200:
                return response
        except requests.exceptions.RequestException as e:
            print(f'Lỗi khi truy cập {url}: {e}. Thử lại lần {attempt + 1}/{max_retries}')
            time.sleep(sleep_time)
    return None
```



## SELENIUM

- Import các thư viện cần thiết

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.common.exceptions import NoSuchElementException
from time import sleep
import pyodbc
```

- Hàm dùng để thu thập dữ liệu từ web

```
def crawl_data(conn, list_link):
    for link in list_link:
        driver.get(link)
        sleep(5)

        try:
            AnimeName = driver.find_element(
                By.CSS_SELECTOR, 'h1.title-name'
            ).text
        except NoSuchElementException:
            AnimeName = None

        try:
            Score = driver.find_element(
                By.CSS_SELECTOR, 'div.score-label'
            ).text
        except NoSuchElementException:
            Score = None

        try:
            Synopsis = driver.find_element(
                By.XPATH, './p[@itemprop="description"]'
            ).text
        except NoSuchElementException:
            Synopsis = None

        # Khởi tạo giá trị mặc định
        AnimeType = Studio = Premiered = Episodes = AnimeStatus = Duration = None
        Rating = Favorites = Ranking = Members = Genres = Popularity = None

        div_elements = driver.find_elements(By.CSS_SELECTOR, 'div.spaceit_pad')
```

- Hàm lấy thông tin từ trang web

```
# Hàm để lấy thông tin từ trang anime
def scrape_anime_list(html_soup):
    for anime in html_soup.select('tr.ranking-list'):
        anime_name = anime.find(class_='fl1 fs14 fw-b anime_ranking_h3').get_text(strip=True)
        print(f'Anime name being scraped: {anime_name}')

        anime_score = anime.select_one('.score-label').text.strip()
        anime_rank = anime.select_one('.top-anime-rank-text').text.strip()

        # Scrape thể loại của anime từ trang chi tiết
        anime_url = anime.select_one('.title .hoverinfo_trigger').get('href')
        anime_detail = fetch_url(anime_url)

        if anime_detail is not None:
            detail_soup = BeautifulSoup(anime_detail.text, 'html.parser')
            genres = ', '.join([genre.text for genre in detail_soup.select('span[itemprop="genre"]')])

            # Scraping additional information
            def get_clean_text(selector):
                element = soup.select_one(selector)
                return re.sub('[^\\d]', '', element.get_text(strip=True)) if element else "N/A"

            # Thực hiện request với cơ chế retry
            r = fetch_url(url)
            if r is not None:
                html_soup = BeautifulSoup(r.text, 'html.parser')

                # Scrape thông tin anime từ trang hiện tại
                scrape_anime_list(html_soup)
```

- Hàm chính để scrape dữ liệu

```
# Bắt đầu quá trình scrape 3 trang đầu tiên
for page in range(180, 240): # Crawl 3 pages
    url = base_url if page == 0 else f'{base_url}?limit={page*50}'
    print('Now scraping page:', url)

    # Thực hiện request với cơ chế retry
    r = fetch_url(url)
    if r is not None:
        html_soup = BeautifulSoup(r.text, 'html.parser')

        # Scrape thông tin anime từ trang hiện tại
        scrape_anime_list(html_soup)

    # Tạm dừng giữa các lần request để tránh bị chặn
    time.sleep(15)
```

- Hàm dùng để thu thập link của các bộ phim Anime

```
def get_anime_links(start_page, end_page, step=50):
    list_link = []
    for page in range(start_page, end_page, step):
        url = f'https://myanimelist.net/topanime.php?limit={page}'
        driver.get(url)
        sleep(5)

        try:
            list_tr = driver.find_elements(
                By.CSS_SELECTOR, 'tr.ranking-list'
            )
            for item in list_tr:
                link = item.find_element(
                    By.CSS_SELECTOR, 'a.hoverinfo_trigger'
                ).get_attribute('href')
                list_link.append(link)
        except NoSuchElementException:
            print('Không tìm được danh sách anime')
            continue

    return list_link
```

- Kết hợp 4 hàm trên lại với nhau để tạo thành 1 chương trình đầy đủ

```
# Main script
if __name__ == "__main__":
    driver = webdriver.Chrome()

    # Lấy danh sách các Liên kết anime từ trang topanime
    list_link = get_anime_links(8900, 8950)

    print(f'Dã thu thập được {len(list_link)} liên kết anime.')

    # Kết nối cơ sở dữ liệu và thu thập dữ liệu chi tiết từ từng Liên kết
    conn = connect_to_sql()
    crawl_data(conn, list_link)
    conn.close()

    driver.quit()
```

## R2: ĐƯA DỮ LIỆU VÀO CƠ SỞ DỮ LIỆU



### KẾT NỐI CƠ SỞ DỮ LIỆU SAU KHI SCRAPING DỮ LIỆU

- Thông tin server để kết nối

```
header = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) \
AppleWebKit/537.36 (KHTML, like Gecko) \
Chrome/126.0.0.0 Safari/537.36 coc_coc_browser/132.0.0'}
```

# Tạo kết nối tới SQL Server

```
params = urllib.parse.quote_plus(
    "DRIVER={ODBC Driver 17 for SQL Server};"
    "SERVER=DESKTOP-6LJ75VT\NHUYXIXI;" # Tên server
    "DATABASE=Anime;" # Tên cơ sở dữ liệu
    "UID=sa;" # Username để đăng nhập SQL Server
    "PWD=ttny01122004;" # Mật khẩu
)
```

- Chuẩn bị dữ liệu và đưa dữ liệu vào SQL server

```
# Chuẩn bị dữ liệu để chèn vào cơ sở dữ liệu
data = {
    'AnimeName': anime_name,
    'Score': float(anime_score) if anime_score.replace('.', '', 1).isdigit() else None,
    'Ranking': int(anime_rank) if anime_rank.isdigit() else None,
    'Genres': genres,
    'AnimeType': animetype,
    'Studio': studio,
    'Premiered': premiered,
    'Synopsis': synopsis,
    'Episodes': int(episodes) if episodes.isdigit() else None,
    'AnimeStatus': status,
    'Duration': duration,
    'Popularity': popularity,
    'Rating': rating,
    'Members': int(members) if members.isdigit() else None,
    'Favorites': int(favorites) if favorites.isdigit() else None
}

# Chèn dữ liệu vào cơ sở dữ liệu
with engine.connect() as connection:
    insert_stmt = insert(anime_table).values(data)
    connection.execute(insert_stmt)
```

- Tạo kết nối đến SQL server

```
# Tạo engine kết nối tới SQL Server
engine = create_engine("mssql+pyodbc://?odbc_connect=%s" % params)
connection = engine.connect()

# Khởi tạo metadata và định nghĩa bảng
metadata = MetaData(bind=engine)

# Định nghĩa bảng AnimeData6
anime_table = Table('AnimeData6', metadata, autoload=True)
```

- Đóng kết nối đến SQL server sau khi đã đưa dữ liệu vào

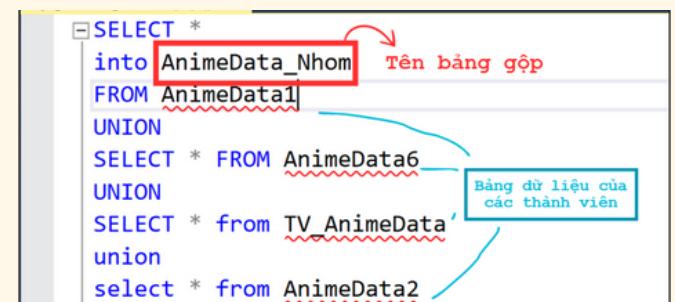
```
# Đóng kết nối SQLAlchemy
engine.dispose()
print("Đã lưu dữ liệu anime vào SQL Server.")
```



### TỔNG HỢP VÀO MỘT CƠ SỞ DỮ LIỆU

Sau khi các thành viên nhóm hoàn thành cào dữ liệu và đưa dữ liệu vào CSDL cá nhân, nhóm sử dụng **SELECT INTO** kết hợp với **UNION** để tạo một bảng mới từ dữ liệu được kết hợp từ 04 bảng dữ liệu.

AnimID	AnimeName	AnimeType	Studio	Premiered	Score	Ranking	Genres	Synopsis
1	Souso no Frieren	TV	Madhouse	Fall 2023	9.33	1	Adventure, Drama, Fantasy, Shounen	During their di...
2	Futilemetal Aheimer: Brotherhood	TV	Bones	Spring 2009	9.09	2	Action, Adventure, Drama, Fantasy, Military, Shoun...	After a horrific...
3	Steins;Gate	TV	White Fox	Spring 2011	9.07	3	Drama, Sci-Fi, Suspense, Psychological, Time Tra...	Eccentric scie...
4	Gintama	TV	Bandai Namco Pictures	Spring 2015	9.06	4	Action, Comedy, Sci-Fi, Gay Humor, Historical, Par...	Gintoki, Shim...
5	Shingeki no Kyojin Season 3 Part 2	TV	Wit Studio	Spring 2019	9.05	5	Action, Drama, Fantasy, Gore, Military, Survival, Thriller	Seeking to ret...
6	Gintama: The Final	Movie	Bandai Namco Pictures	N/A	9.04	6	Action, Comedy, Drama, Sci-Fi, Gay Humor, History...	After a year...
7	Gintama*	TV	Sunrise	Spring 2011	9.03	7	Action, Comedy, Sci-Fi, Gay Humor, Historical, Par...	After a one-ye...
8	Hunter x Hunter (2011)	TV	Madhouse	Fall 2011	9.03	8	Action, Adventure, Fantasy, Shounen	Hunters dead...
9	Monogatari Series: Off & Monster Season	ONA	Shelf	N/A	9.03	9	Comedy, Mystery, Supernatural, Vampire	Koyomi Ararai...
10	Gintama* Enchousen	TV	Sunrise	Fall 2012	9.02	10	Action, Comedy, Sci-Fi, Gay Humor, Historical, Par...	White Gintoki...
11	Bleach: Sennen Kessen-hen	TV	Pierrot	Fall 2022	9.01	11	Action, Adventure, Fantasy, Shounen	Substitute So...
12	Ginga Eiyuu Denjutsu	OVA	K-Factory, Kite Film Makaia Studio	N/A	9.01	12	Drama, Sci-Fi, Adult Cast, Military, Space	The 150-year...
13	Kaguya-sama wa Kokurasetai: Ultra Romantic	TV	A-1 Pictures	Spring 2022	9	13	Comedy, Romance, School, Seinen	The elite men...
14	Gintama	TV	Bandai Namco Pictures	Winter 2017	8.98	14	Action, Comedy, Sci-Fi, Gay Humor, Historical, Par...	After joining th...
15	Space Basket: The Final	TV	TMS Entertainment	Spring 2021	8.98	15	Comedy, Drama, Sports, Superhero, Shounen	One day, I will...
16	Classmate After Story	TV	Media Information	Spring 2008	8.93	16	Drama, Romance, Superhero	Tomoko Okada...
17	Gintama	TV	Sunrise	Spring 2006	8.93	17	Action, Comedy, Sci-Fi, Gay Humor, Historical, Par...	Edo is a city...
18	Koe no Katachi	Movie	Kyoto Animation	N/A	8.93	18	Action, Adventure, Drama, Shounen	As a wild youth...
19	3-gatsu no Lion 2nd Season	TV	Shelf	Fall 2017	8.91	19	Drama, Children, Strategy Game, Seinen	New in his se...



Nhóm thu được 01 bảng dữ liệu gồm đầy đủ các trường dữ liệu theo nhu cầu của nhóm và thỏa mãn yêu cầu có nhiều hơn **10.000** dòng dữ liệu

# R3: TIỀN XỬ LÝ VÀ PHÂN CHIA CẤU TRÚC CHO CƠ SỞ DỮ LIỆU



## TIỀN XỬ LÝ DỮ LIỆU

### Chuyển định dạng Duration thành phút

```

CREATE TRIGGER Convert_Duration_Trigger
ON [dbo].[AnimeTong]
AFTER INSERT, UPDATE
AS
BEGIN
    -- Chuyển các giá trị khác unknown thành phút
    UPDATE [dbo].[AnimeTong]
    SET Duration_min =
        CASE
            -- Trường hợp có định dạng phút cho mỗi tập
            WHEN Duration LIKE '#min. per ep.%' THEN
                CAST(SUBSTRING(Duration, 1, CHARINDEX(' min.', Duration) - 1) AS INT)
            -- tương tự với các trường hợp còn lại
            -- Trường hợp là 'unknown', giữ nguyên giá trị
            WHEN Duration = 'unknown' THEN NULL -- Giữ nguyên giá trị 'unknown'
        END
    WHERE Duration IS NOT NULL
    AND AnimeID IN (SELECT AnimeID FROM INSERTED); -- Chỉ cập nhật các hàng có trong bảng INSERT

```

### Xử lý giá trị thiếu ở cột Premiered, Studio, Episodes

```

CREATE TRIGGER Clean_Trigger
ON [dbo].[AnimeTong]
AFTER INSERT, UPDATE
AS
BEGIN
    --1. Cột Premiered: Thay null/"" bằng "N/A"
    UPDATE [dbo].[AnimeTong]
    SET Premiered = 'N/A'
    WHERE Premiered IS NULL OR Premiered = ''
    AND AnimeID IN (SELECT AnimeID FROM INSERTED)
    --2. Cột Studio: Thay null/ "None found, add some" bằng "N/A"
    UPDATE [dbo].[AnimeTong]
    SET Studio = 'N/A'
    WHERE Studio IS NULL OR Studio = 'None found, add some'
    AND AnimeID IN (SELECT AnimeID FROM INSERTED)
    --3. Cột Episodes: Thay null bằng 1
    UPDATE [dbo].[AnimeTong]
    SET Episodes = 1
    WHERE Episodes IS NULL
    AND AnimeID IN (SELECT AnimeID FROM INSERTED)
END;

```

### Kiểm tra tính hợp lệ của dữ liệu

```

CREATE TRIGGER CheckAnimeData
ON AnimeData
AFTER INSERT, UPDATE
AS
BEGIN
    -- Kiểm tra Score phải nằm trong khoảng từ 0 đến 10
    IF EXISTS (SELECT 1 FROM inserted WHERE Score < 0 OR Score > 10)
    BEGIN
        RAISERROR('Score must be between 0 and 10', 16, 1)
        ROLLBACK TRANSACTION
        RETURN
    END

    -- Kiểm tra Ranking phải là số dương
    IF EXISTS (SELECT 1 FROM inserted WHERE Ranking <= 0)
    BEGIN
        RAISERROR('Ranking must be a positive number', 16, 1)
        ROLLBACK TRANSACTION
        RETURN
    END

    -- Kiểm tra Members và Favorites không được âm
    IF EXISTS (SELECT 1 FROM inserted WHERE Members < 0 OR Favorites < 0)
    BEGIN
        RAISERROR('Members and Favorites must be non-negative', 16, 1)
        ROLLBACK TRANSACTION
        RETURN
    END

```

### Xóa những dòng trống

```

CREATE PROCEDURE XoaDongTrong
AS
BEGIN
    -- Xóa các dòng có AnimeID nhưng tất cả các cột còn lại đều trống hoặc NULL
    DELETE FROM AnimeData
    WHERE
        -- Cột AnimeName trống hoặc NULL
        (AnimeName IS NULL OR LTRIM(RTRIM(AnimeName)) = '');

```

### Xóa các dòng trùng lặp

```

CREATE PROCEDURE RemoveDuplicateRows
AS
BEGIN
    -- Xóa các dòng trùng lặp dựa trên AnimeID, giữ lại bản ghi đầu tiên
    WITH DuplicateRows AS (
        SELECT
            AnimeID,
            ROW_NUMBER() OVER(PARTITION BY AnimeID ORDER BY (SELECT NULL)) AS RowNum
        FROM
            dbo.AnimeData
    )
    DELETE FROM DuplicateRows
    WHERE RowNum > 1; -- Chỉ xóa các bản ghi có RowNum lớn hơn 1 (tức là bản ghi trùng lặp)

```

### Xóa các khoảng trắng

```

CREATE PROCEDURE XoaKhoangTrang
@TableName NVARCHAR(255), -- Tên bảng
@ColuAnimeDataName NVARCHAR(255) -- Tên cột cần loại bỏ khoảng trắng
AS
BEGIN
    DECLARE @SQL NVARCHAR(MAX);
    -- Câu lệnh SQL động để cập nhật tất cả các bản ghi trong cột đã chỉ định và loại bỏ khoảng trắng ở đầu và cuối
    SET @SQL = 'UPDATE ' + @TableName +
        ' SET ' + @ColuAnimeDataName + ' = LTRIM(RTRIM(' + @ColuAnimeDataName + '))';
    -- Thực thi câu lệnh SQL động
    EXEC sp_executesql @SQL;
END;

```

### Xóa chú thích cột Rating

```

CREATE PROCEDURE RemoveRatingComments
AS
BEGIN
    -- Cập nhật cột Rating, giữ lại các giá trị chính và xóa phần chú thích
    UPDATE AnimeData
    SET Rating = CASE
        WHEN Rating LIKE 'G%' THEN 'G'
        WHEN Rating LIKE 'PG-13%' THEN 'PG-13'
        WHEN Rating LIKE 'R+' THEN 'R+'
        WHEN Rating LIKE 'PG%' THEN 'PG'
        WHEN Rating LIKE 'None%' THEN 'None'
        Else 'R-17+'
    END
    WHERE Rating LIKE '%-%'
END

```

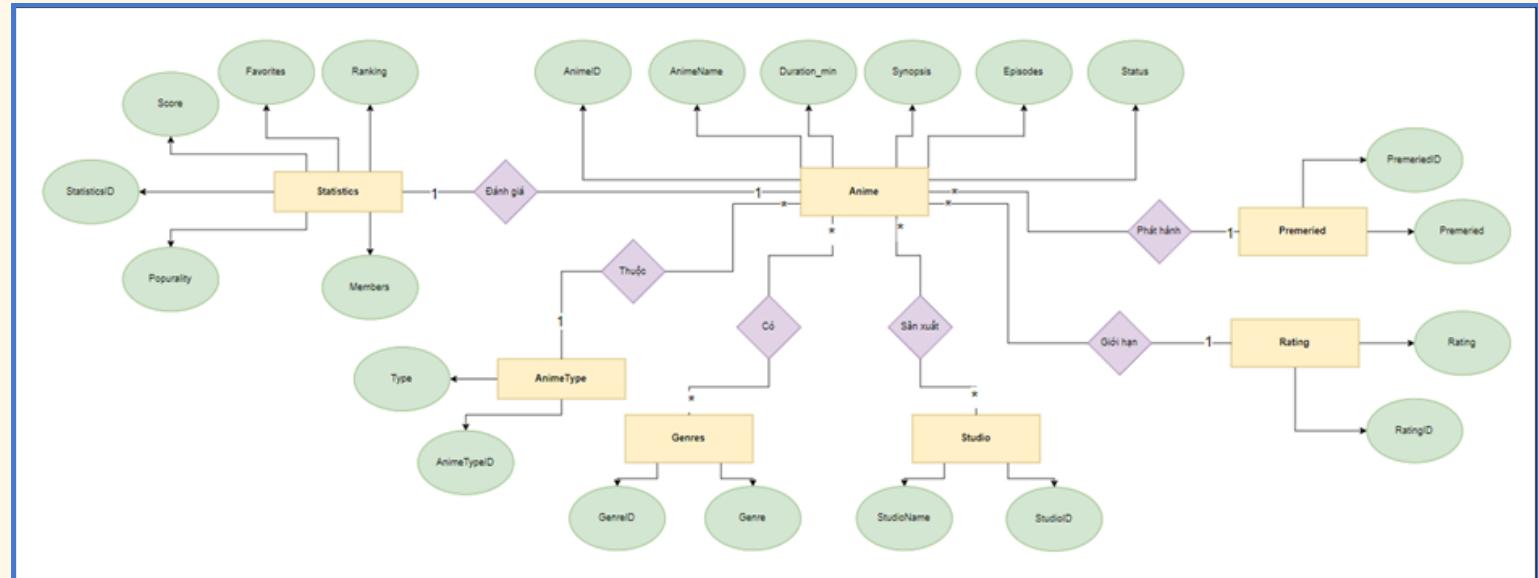




## PHÂN CHIA CẤU TRÚC CHO CƠ SỞ DỮ LIỆU

### Mô hình ER, DR

Với dữ liệu ban đầu sau khi lấy về là 1 bảng tổng hợp thì nhóm sẽ tổ chức lại cấu trúc dữ liệu:



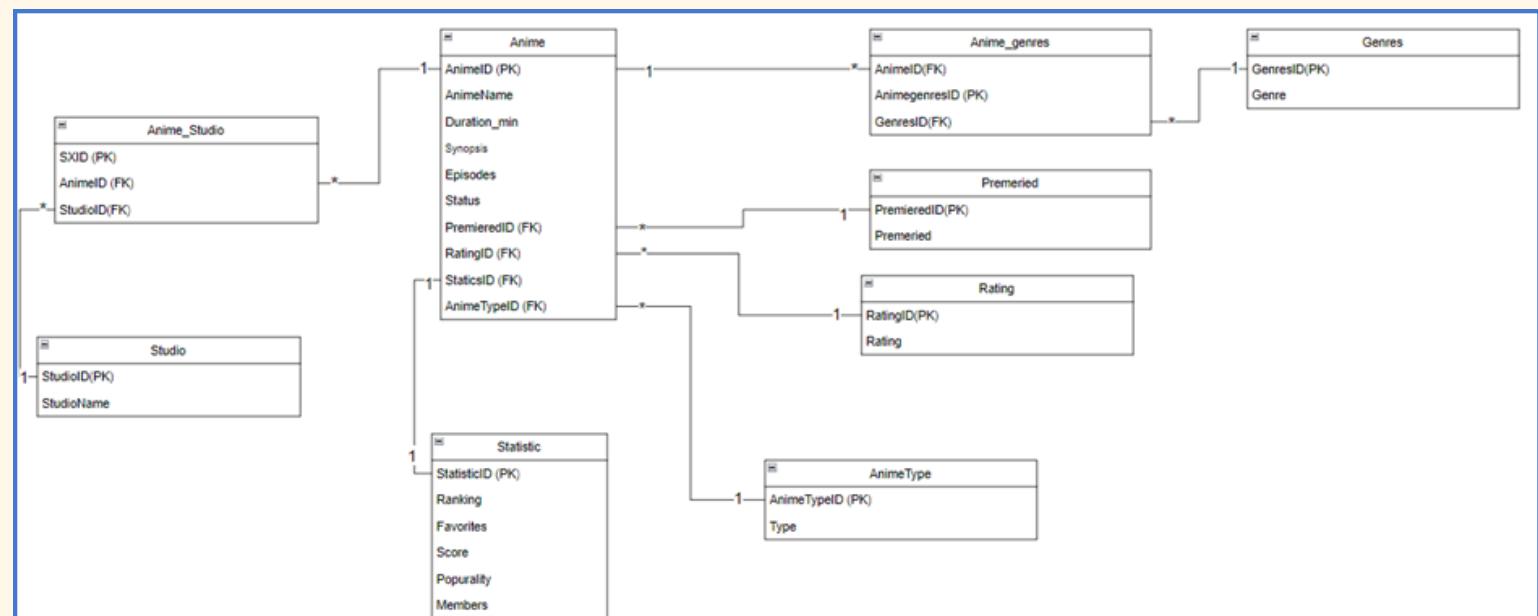
Bảng Anime sẽ bao gồm các thông tin về các bộ phim Anime như Mô tả, Thời lượng,...

- Các cột Premiered, AnimeType, Rating sẽ được tách làm bảng riêng và nối với bảng Anime theo mối quan hệ 1-nhiều.

- Bảng Studio và Genres sẽ được nối với bảng Anime theo mối quan hệ nhiều-nhiều vì một bộ phim có nhiều thể loại và nhiều xưởng sản xuất, và ngược lại.

- Các đánh giá của người dùng là dữ liệu dễ thay đổi nên sẽ được tách làm bảng Statistics có mối quan hệ 1-1 với bảng Anime.

Sau khi có Sơ đồ Thực thể - Mối quan hệ thì nhóm đã tạo được Sơ đồ Dữ liệu - Mối quan hệ :

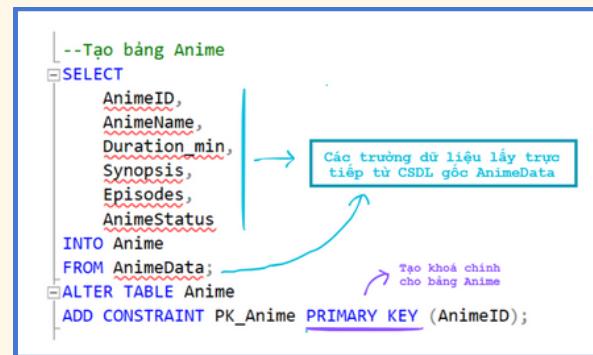




## PHÂN CHIA CẤU TRÚC CHO CƠ SỞ DỮ LIỆU

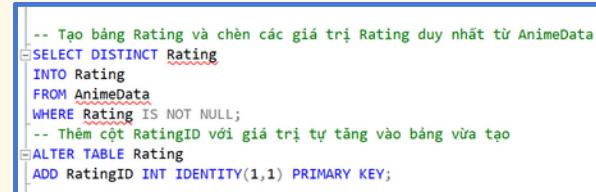
### Thực hiện tách bảng và kết quả

Theo sơ đồ ER đã trình bày, đầu tiên nhóm tiến hành tạo bảng chính: **Anime**

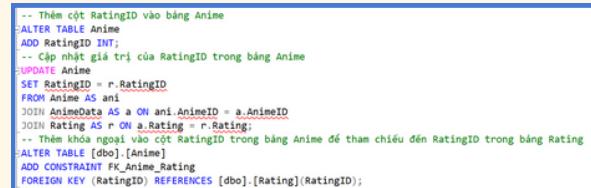


Tiếp theo, nhóm tiến hành tạo các bảng phụ, bao gồm **Rating**, **Statistic**, **Premiered** và **AnimeType**. Các bảng này được xây dựng với mục đích phân tách và quản lý dữ liệu theo từng khía cạnh cụ thể.

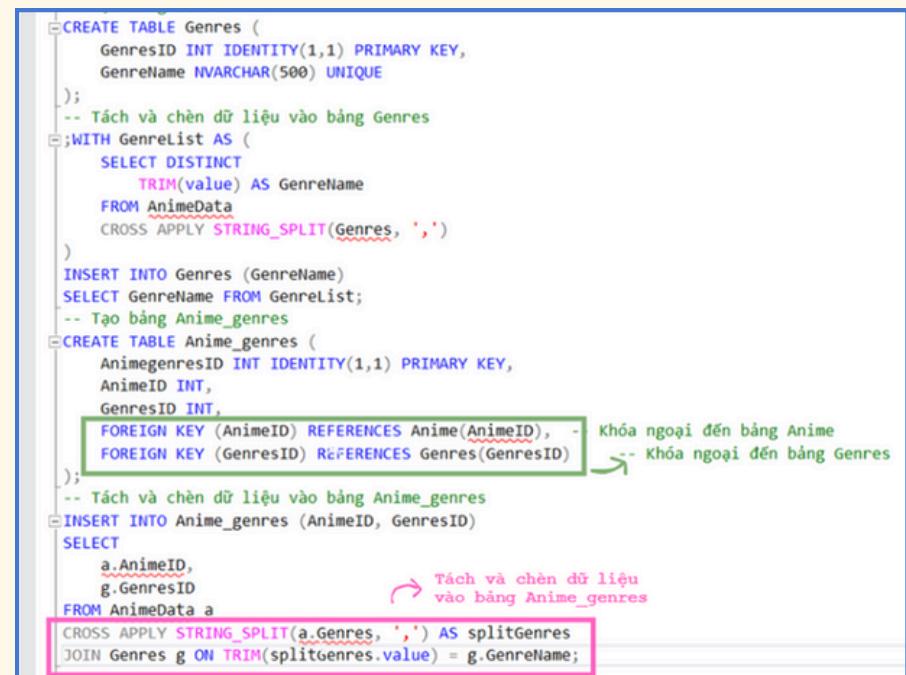
Ví dụ với bảng Rating (tương tự các bảng còn lại)



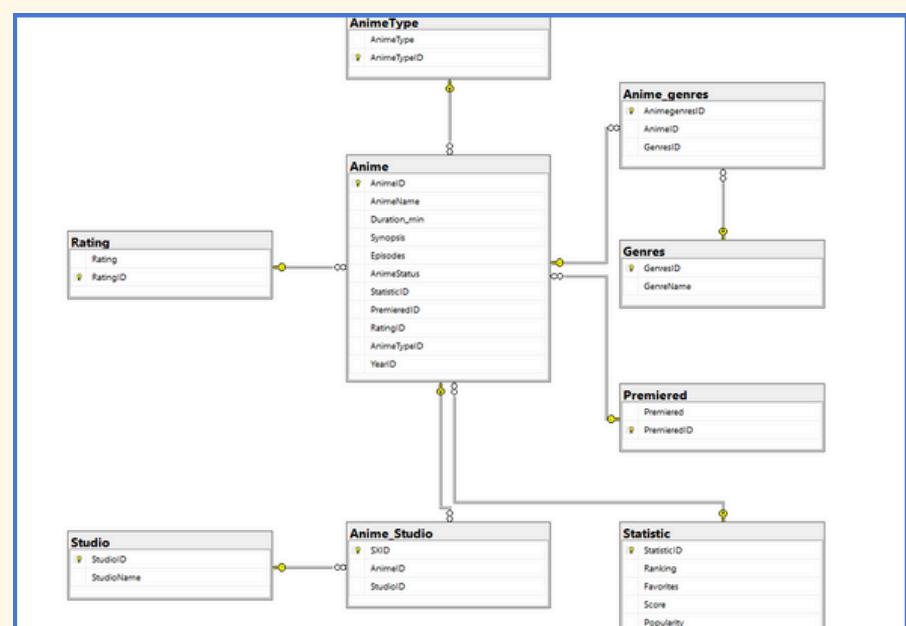
Mỗi bảng phụ đều được định nghĩa với một khóa chính (**RatingID**, **StatisticID**, **PremieredID**, **AnimeTypeID**) đóng vai trò là khóa ngoại tham chiếu đến khóa chính của bảng trung tâm Anime, nhằm đảm bảo tính toàn vẹn dữ liệu và mối quan hệ chặt chẽ giữa các bảng.



Sau đó, nhóm tạo bảng Genres và Studio với bảng phụ Anime\_genres và Anime\_Studio để nối bảng Genres, Studio và bảng chính Anime



Kết quả: Nhóm có CSDL hoàn chỉnh



## R4: BACKUP DỮ LIỆU



### ĐỊNH NGHĨA BACKUP VÀ LỰA CHỌN LOẠI BACKUP

**B**ackup dữ liệu (sao lưu dữ liệu): là quá trình tạo ra một bản sao của dữ liệu và lưu trữ ở một vị trí khác an toàn.

Với CSDL Anime của nhóm: ít thay đổi, không quá lớn và ưu nhược điểm của hai loại backup, nhóm sử dụng kết hợp 2 loại backup, cụ thể:

- Full Backup: thực hiện để sao lưu CSDL lần đầu tiên và **hàng tháng** vào lúc 3h sáng của ngày đầu tiên trong tháng
- Differential Backup: thực hiện **hàng tuần** vào 4h sáng của ngày chủ nhật

Full backup	Differential Backup
Sao lưu <b>toàn bộ</b> dữ liệu của database	Sao lưu <b>những thay đổi</b> kể từ lần full backup gần nhất
Thường dùng định kỳ ( <b>hàng tháng</b> , hàng tuần)	Thường dùng xen kẽ giữa các lần full backup ( <b>hàng tuần</b> , hàng ngày)
Sao lưu tất cả dữ liệu, đơn giản khi khôi phục lại dữ liệu	Nhanh, tiết kiệm dung lượng. Phù hợp với những cơ sở dữ liệu cần backup thường xuyên
Mất nhiều thời gian và dung lượng lưu trữ	Phục hồi phức tạp hơn và phụ thuộc vào full backup

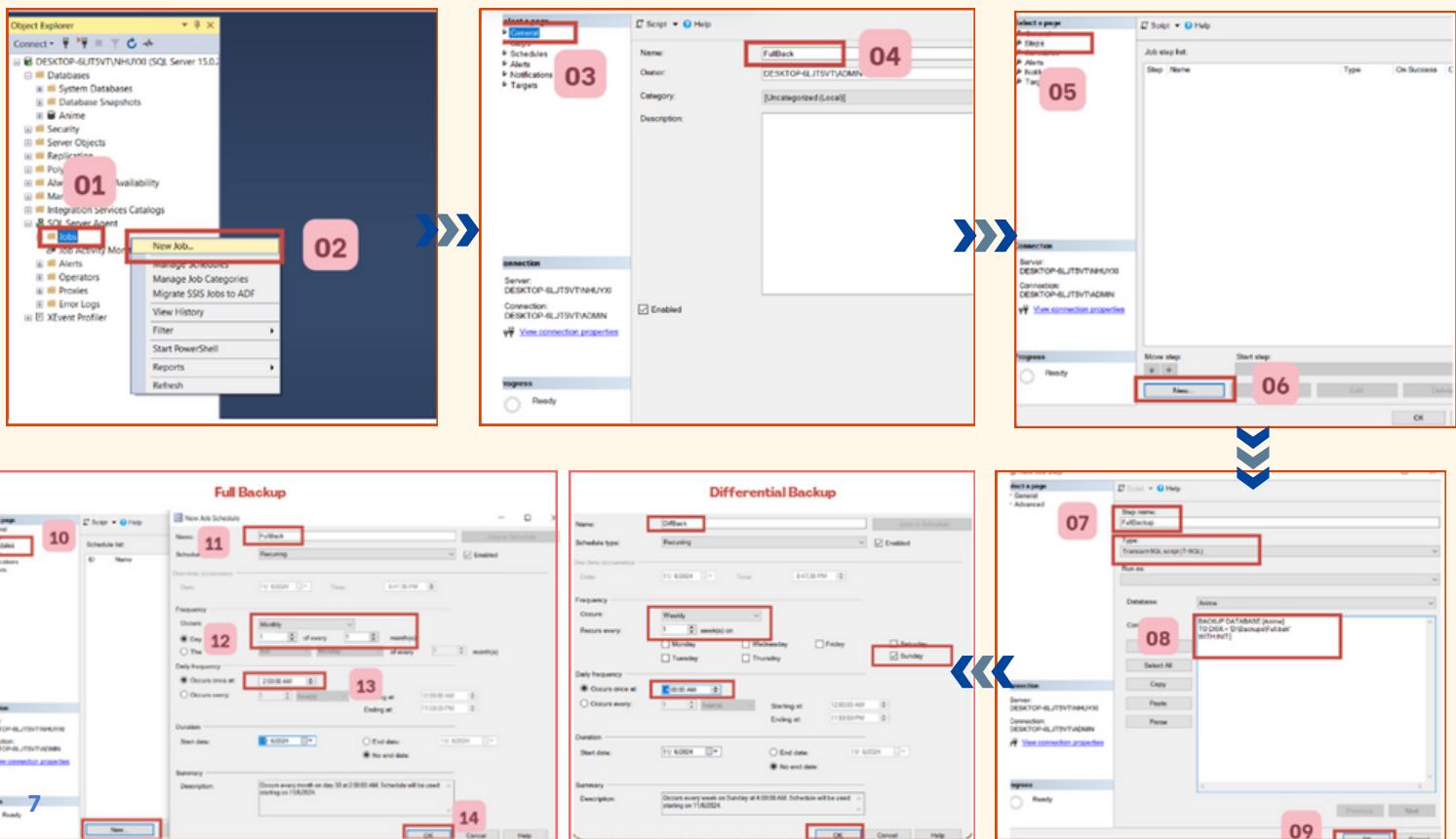


### QUY TRÌNH BACKUP



### BẮNG THAO TÁC

Tạo backup từ Job của SQL Server Agent:



# QUY TRÌNH THỰC HIỆN BACKUP BẰNG T-SQL



## Bước 1: Thực hiện Backup Database:

Full Backup	Differential Backup
<pre>--Backup full BACKUP DATABASE [Anime_Org] TO DISK = N'D:\Backup\DatabaseName Full.bak' WITH INIT, COMPRESSION, STATS = 10;</pre>	<pre>--Backup diffrent BACKUP DATABASE [Anime_Org] TO DISK = N'D:\Backup\DatabaseName Diff.bak' WITH DIFFERENTIAL, INIT, COMPRESSION, STATS = 10;</pre>

## Bước 2: Tạo job

-- Tạo job	-- Tạo job
<pre>EXEC sp_add_job @job_name = N'Monthly_Full_Backup';</pre>	<pre>EXEC sp_add_job @job_name = N'Weekly_Differential_Backup';</pre>

## Bước 3: Thêm các bước backup vào job

-- Thêm bước backup vào job	-- Thêm bước backup vào job
<pre>EXEC sp_add_jobstep @job_name = N'Monthly_Full_Backup', @step_name = N'Full Backup Step', @subsystem = N'TSQL', @command = N'BACKUP DATABASE [Anime_Org] TO DISK = N'D:\Backup\DatabaseName_Full.bak' WITH INIT, COMPRESSION, STATS = 10;', @database_name = N'Anime_Org';</pre>	<pre>EXEC sp_add_jobstep @job_name = N'Weekly_Differential_Backup', @step_name = N'Differential Backup Step', @subsystem = N'TSQL', @command = N'BACKUP DATABASE [Anime_Org] TO DISK = N'D:\Backup\DatabaseName_Diff.bak' WITH DIFFERENTIAL, INIT, COMPRESSION, STATS = 10;'; @database_name = N'Anime_Org';</pre>

## Bước 4: Lập lịch cho job

-- Tạo lịch trình chạy vào ngày 1 mỗi tháng lúc 3:00 AM	-- Thêm lịch chạy job vào mỗi Chủ nhật lúc 4h sáng
<pre>EXEC sp_add_jobschedule @job_name = N'Monthly_Full_Backup', @name = N'Weekly Schedule', @freq_type = 4, -- Định kỳ (Scheduled) @freq_interval = 30, -- sau 30 ngày kể từ ngày backup gần nhất @freq_recurrence_factor = 1, -- Lặp lại mỗi tháng @active_start_time = 30000; -- 3:00 AM</pre>	<pre>EXEC sp_add_jobschedule @job_name = N'Weekly_Differential_Backup', @name = N'Weekly Schedule', @freq_type = 8, -- Chạy hàng tuần @freq_interval = 1, -- Chủ nhật (1: Chủ nhật) @freq_recurrence_factor = 1, -- Lặp lại mỗi tuần @active_start_time = 40000; -- 4:00 AM</pre>

## Bước 5: Liên kết job với SQL Server Agent

-- Liên kết job với SQL Server Agent	-- Liên kết job với SQL Server Agent
<pre>EXEC sp_add_jobserver @job_name = N'Monthly_Full_Backup';</pre>	<pre>EXEC sp_add_jobserver @job_name = N'Weekly_Differential_Backup';</pre>

## Bước 6: Kiểm tra job

Name	Enabled	Status	Last Run	Next Run
Job Monthly_Full_Backup	Yes	Idle	Unknown	never
Job Weekly_Differential_Backup	Yes	Idle	Success	11/6/2024 2:10:35

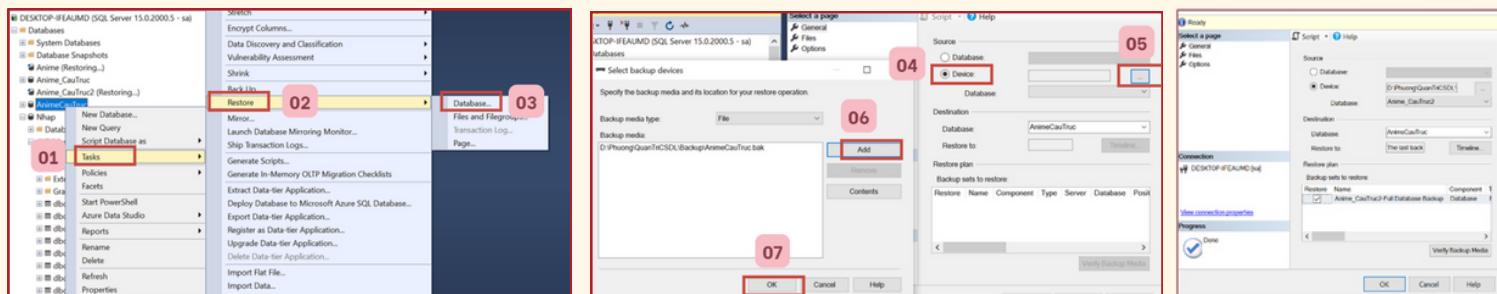
**ANIME**

MAKES ME HAPPY WHEN NO ONE ELSE DOES



# RESTORE DỮ LIỆU

Sau khi thực hiện backup thì ta có thể khôi phục lại dữ liệu từ file backup.bak đó như sau:



## R5: PHÂN QUYỀN TRONG SQL SERVER MANAGEMENT



### PHÂN QUYỀN CHO HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU BẰNG T-SQL



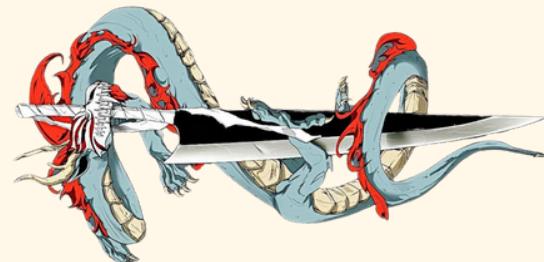
#### XÁC ĐỊNH CÁC QUYỀN CHO TỪNG ĐỐI TƯỢNG

	Quyền	Admin	DE	DA
<b>CẤU TRÚC</b>	Create/Delete table	✓		
	Alter table	✓		
<b>DỮ LIỆU</b>	Backup	✓	✓	
	Create Funtion/Trigger/Procedure	✓	✓	
	Exec Funtion/Trigger/Procedure	✓	✓	✓
	Update	✓	✓	
	Delete	✓	✓	
	Select	✓	✓	✓
	View	✓	✓	✓
	Insert	✓	✓	

**Admin:** Quản trị viên hệ thống, toàn quyền quản lý.

**DE (Data Engineer):** Kỹ sư dữ liệu, tập trung vào cấu trúc và luồng dữ liệu.

**DA (Data Analyst):** Nhà phân tích dữ liệu, chủ yếu thao tác và phân tích dữ liệu sẵn có.



### QUY TRÌNH PHÂN QUYỀN TRONG HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER



#### PHÂN QUYỀN CHO CƠ SỞ DỮ LIỆU BẰNG T-SQL

##### Bước 1: Tạo tài khoản

###### Admin

```
-- Tạo login và user cho Admin
CREATE LOGIN AdminLogin WITH PASSWORD = 'Anime@admin';
USE Anime;
CREATE USER AdminUser FOR LOGIN AdminLogin;
```

###### DE

```
-- Tạo login và user cho Data Engineer
CREATE LOGIN DE_Login WITH PASSWORD = 'Anime@DE';
USE Anime;
CREATE USER DE_User FOR LOGIN DE_Login;
```

###### DA

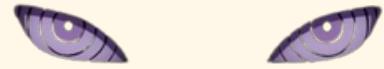
```
-- Tạo login và user cho Data Analyst
CREATE LOGIN DA_Login WITH PASSWORD = 'Anime@DA';
USE Anime;
CREATE USER DA_User FOR LOGIN DA_Login;
```

## Bước 2: Cấp quyền cho Admin

```
-- Cấp quyền sysadmin cho Admin (tất cả quyền trên server)
ALTER SERVER ROLE sysadmin ADD MEMBER AdminLogin;

-- Cấp quyền db_owner cho Admin trong cơ sở dữ liệu cụ thể
USE Anime;
ALTER ROLE db_owner ADD MEMBER AdminUser;

-- Cấp quyền backup và phục hồi cơ sở dữ liệu
GRANT BACKUP DATABASE TO AdminUser;
GRANT BACKUP LOG TO AdminUser;
```



(The understand pass, you must know padin)

## Bước 2: Cấp quyền cho DE

```
-- Cấp quyền SELECT, INSERT, UPDATE, DELETE cho tất cả các bảng
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Anime TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Anime_genres TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Anime_Studio TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.AnimeType TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Genres TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Premiered TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Rating TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Statistic TO DE_User;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.Studio TO DE_User;
--Cấp quyền tạo trigger/function/procedure
GRANT CREATE TRIGGER TO DE_User;
GRANT CREATE FUNCTION TO DE_User;
GRANT CREATE PROCEDURE TO DE_User;

-- Cấp quyền ALTER cho tất cả các bảng
GRANT ALTER ON dbo.Anime TO DE_User;
GRANT ALTER ON dbo.Anime_genres TO DE_User;
GRANT ALTER ON dbo.Anime_Studio TO DE_User;
GRANT ALTER ON dbo.AnimeType TO DE_User;
GRANT ALTER ON dbo.Genres TO DE_User;
GRANT ALTER ON dbo.Premiered TO DE_User;
GRANT ALTER ON dbo.Rating TO DE_User;
GRANT ALTER ON dbo.Statistic TO DE_User;
GRANT ALTER ON dbo.Studio TO DE_User;
GRANT CREATE TRIGGER TO DE_User;

-- Cấp quyền EXECUTE cho stored procedures nếu có
GRANT EXECUTE ON dbo.ETL_StoredProcedure TO DE_User; -- Thay 'ETL_StoredProcedure'
--Cấp quyền backup
GRANT BACKUP DATABASE TO DE_User;
GRANT BACKUP LOG TO DE_User;
```



## Bước 2: Cấp quyền cho DA

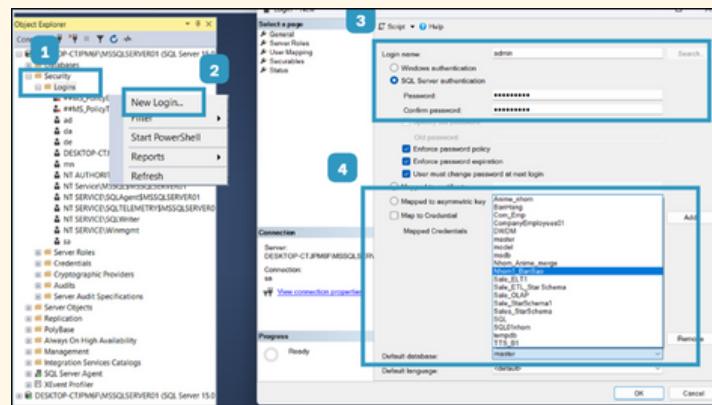
```
-- tạo role
USE Anime;
CREATE ROLE DataAnalystRole;
-- Cấp quyền SELECT cho Role trên tất cả các bảng trừ AnimeData
GRANT SELECT ON dbo.Anime TO DataAnalystRole;
GRANT SELECT ON dbo.Anime_genres TO DataAnalystRole;
GRANT SELECT ON dbo.Anime_Studio TO DataAnalystRole;
GRANT SELECT ON dbo.AnimeType TO DataAnalystRole;
GRANT SELECT ON dbo.Genres TO DataAnalystRole;
GRANT SELECT ON dbo.Premiered TO DataAnalystRole;
GRANT SELECT ON dbo.Rating TO DataAnalystRole;
GRANT SELECT ON dbo.Statistic TO DataAnalystRole;
GRANT SELECT ON dbo.Studio TO DataAnalystRole;

-- Cấp quyền EXECUTE cho role trên các stored procedure / không
GRANT EXECUTE ON dbo.StoredProcedure1 TO DataAnalystRole;
--Cấp quyền tạo view
GRANT CREATE VIEW TO DataAnalystRole;
```

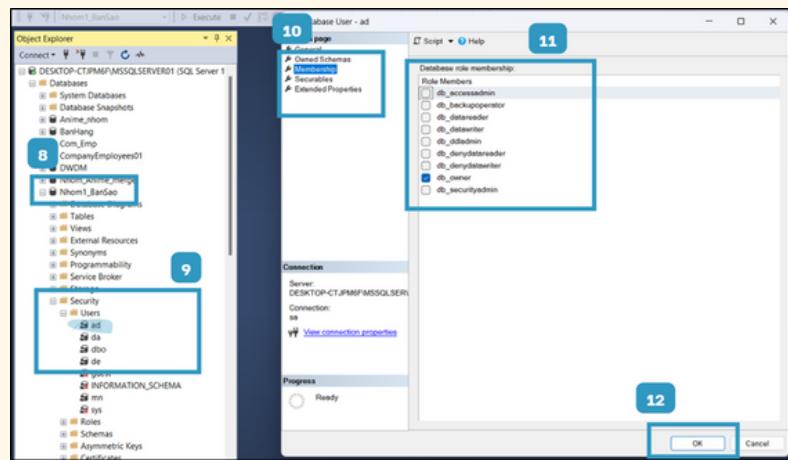


# PHÂN QUYỀN CHO HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU BẰNG THAO TÁC

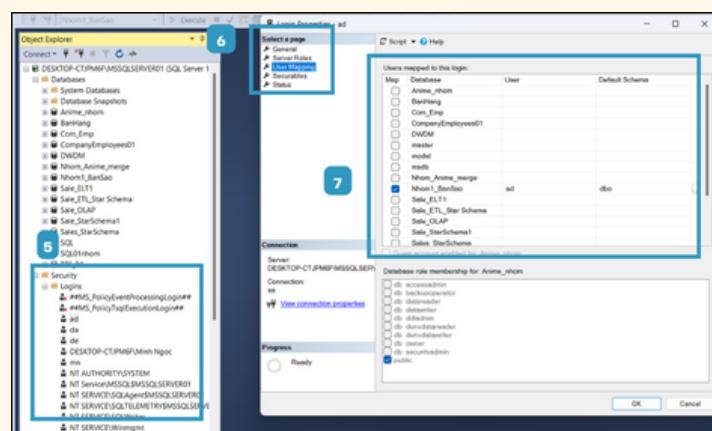
## Bước 1: Tạo tài khoản



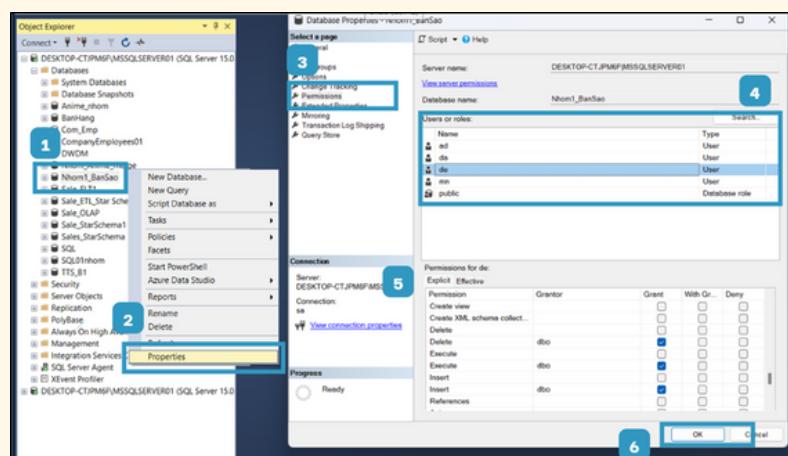
## Bước 2: Cấp quyền cho Admin



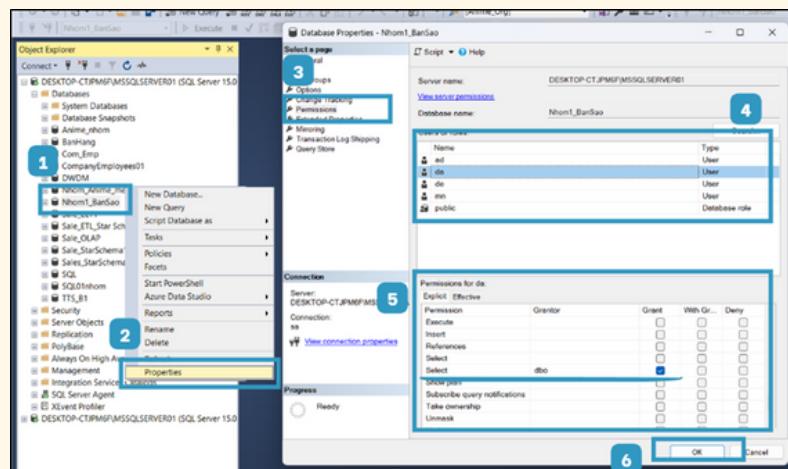
## Bước 1: Ánh xạ đến database



## Bước 2: Cấp quyền cho DE



## Bước 2: Cấp quyền cho DA





## Admin

**DESKTOP-IFEAUDM (SQL Server 15.0.2000.5 - Admin)**

```
-- Tạo bảng mới
CREATE TABLE TestTable (ID INT, Name NVARCHAR(50));
Commands completed successfully.

Completion time: 2024-11-08T10:42:13.3218216+07:00
```

```
-- Thêm dữ liệu
INSERT INTO TestTable (ID, Name) VALUES (1, 'Test');
(1 row affected)

Completion time: 2024-11-08T10:43:39.6559781+07:00
```

```
-- Xóa bảng
DROP TABLE TestTable; Commands completed successfully.

Completion time: 2024-11-08T10:45:29.3926692+07:00
```



DA

**DESKTOP-IFEAUDM (SQL Server 15.0.2000.5 - DA)**

```
-- Tạo bảng mới
CREATE TABLE DE_Table (ID INT, Description NVARCHAR(100));
Msg 262, Level 14, State 1, Line 2
CREATE TABLE permission denied in database 'Anime_CauTruc'.

Completion time: 2024-11-08T10:57:56.2799852+07:00
```

```
-- Thêm dữ liệu
INSERT INTO DE_Table (ID, Description) VALUES (1, 'DE Test');

1 row affected

Completion time: 2024-11-08T11:10:22.0386091+07:00
```

```
-- Sửa dữ liệu
UPDATE DE_Table SET Description = 'Updated Test' WHERE ID = 1;

(1 row affected)

Completion time: 2024-11-08T11:12:39.0284669+07:00
```



DE

**DESKTOP-IFEAUDM (SQL Server 15.0.2000.5 - DE)**

```
-- Tạo bảng mới
CREATE TABLE DE_Table (ID INT, Description NVARCHAR(100));
Msg 262, Level 14, State 1, Line 2
CREATE TABLE permission denied in database 'Anime_CauTruc'.

Completion time: 2024-11-08T10:57:56.2799852+07:00
```

```
-- Thêm dữ liệu
INSERT INTO DE_Table (ID, Description) VALUES (1, 'DE Test');

1 row affected

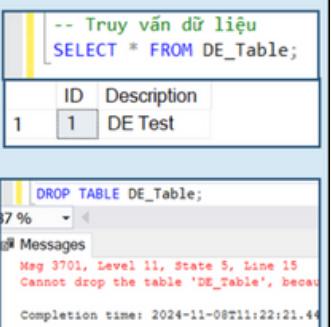
Completion time: 2024-11-08T11:10:22.0386091+07:00
```

```
-- Sửa dữ liệu
UPDATE DE_Table SET Description = 'Updated Test' WHERE ID = 1;

(1 row affected)

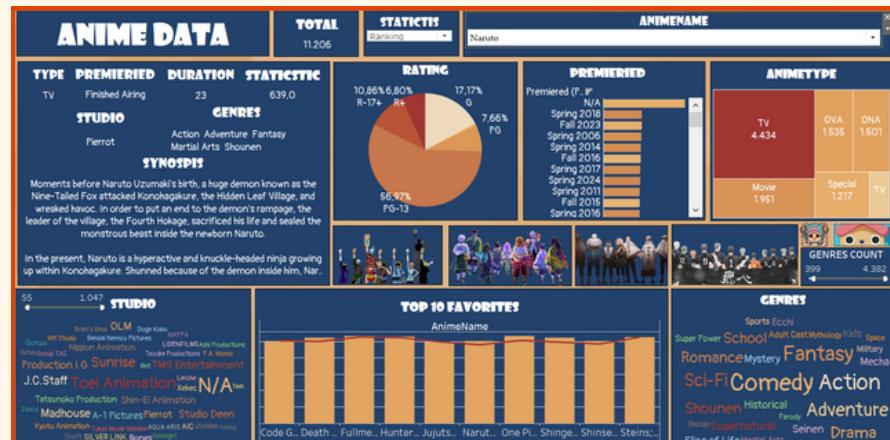
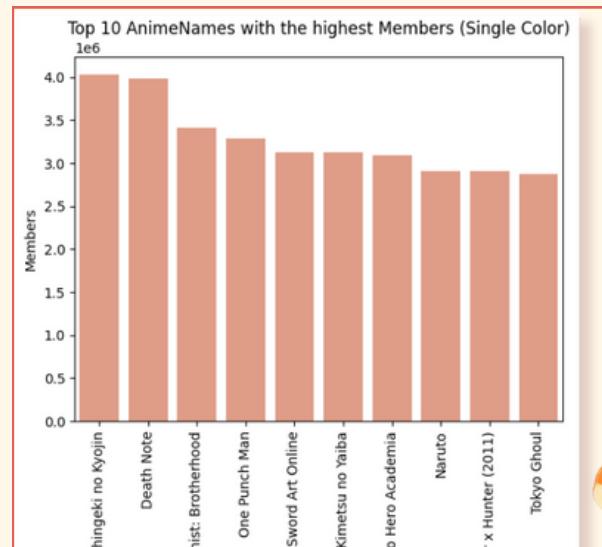
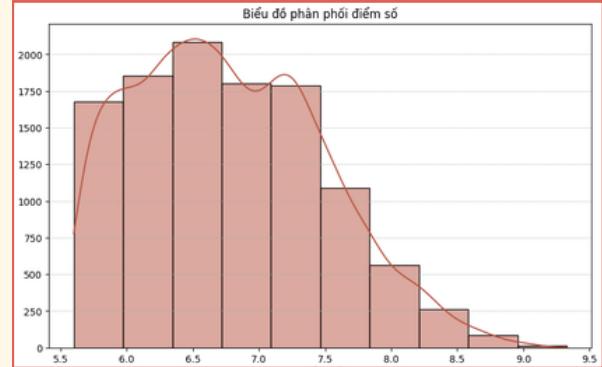
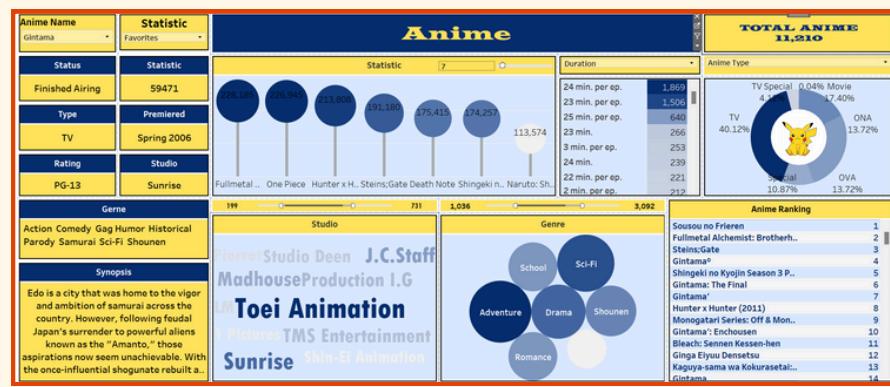
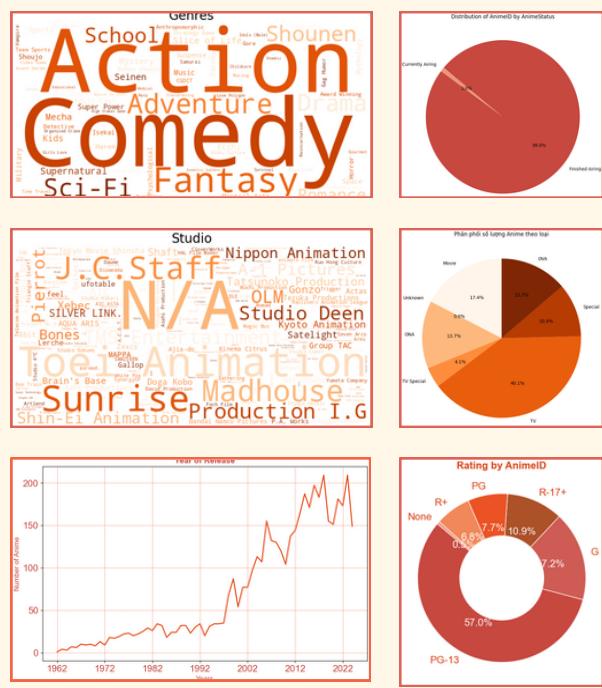
Completion time: 2024-11-08T11:12:39.0284669+07:00
```



# R6: TRỰC QUAN HOÁ DỮ LIỆU



# ANIME DATA



# R7: XÂY DỰNG HỆ THỐNG ĐỀ XUẤT PHIM



## Ý TƯỞNG

- Sử dụng TF-IDF Vectorizer để biến đổi dữ liệu văn bản thành vectơ đặc trưng.
- Sử dụng Cosine Similarity để đo lường mức độ giống nhau giữa các bộ phim.
- Sắp xếp và chọn ra những bộ phim có độ tương đồng cao nhất để đề xuất.



## THỰC HIỆN BẰNG THUẬT TOÁN

- Thực hiện kết nối đến cơ sở dữ liệu

```
# Kết nối tới SQL Server
conn = pyodbc.connect(
    'DRIVER={ODBC Driver 17 for SQL Server};'
    'SERVER=DESKTOP-IFEAUDM;'
    'DATABASE=Anime_CauTruc;'
    'UID=sa;'
    'PWD=3110'
)
# Truy vấn dữ liệu
query = """
SELECT a.AnimeName, a.Synopsis,
       a.Episodes, a.AnimeStatus,
       a.Duration_min, g.GenreName,
       s.StudioName, r.Rating,
       p.Premiered, t.AnimeType,
       st.Ranking, st.Popularity
FROM Anime a
JOIN Anime_genres ag ON a.AnimeID = ag.AnimeID
JOIN Genres g ON ag.GenresID = g.GenresID
JOIN Anime_Studio asd ON a.AnimeID = asd.AnimeID
JOIN Studio s ON asd.StudioID = s.StudioID
JOIN Rating r ON a.RatingID = r.RatingID
JOIN Statistic st ON a.StatisticID = st.StatisticID
JOIN Premiered p ON a.PremieredID = p.PremieredID
JOIN AnimeType t ON a.AnimeTypeID = t.AnimeTypeID
"""
df_f = pd.read_sql_query(query, conn)

# Đóng kết nối
conn.close()
```

- Gộp Genres và Studio vào 1 cột

```
# Gộp các giá trị cho GenreName và StudioName
df = df_f.groupby([
    'AnimeName', 'Synopsis', 'Episodes', 'AnimeStatus', 'Duration_min',
    'Rating', 'Premiered', 'AnimeType', 'Ranking', 'Popularity'], as_index=False
).agg({
    'GenreName': lambda x: ', '.join(x.unique()),
    'StudioName': lambda x: ', '.join(x.unique())
})
```

- Tiền xử lý dữ liệu

```
import re
df['Genres'] = df['Genres'].apply(lambda x: re.sub(r'[^\w\s]', '', x.lower()) if isinstance(x, str) else '')
# Tạo cột mô tả tổng hợp
df['Description'] = df['Studio'] + ' ' + \
    df['Premiered'] + ' ' + \
    df['Ranking'].astype(str) + ' ' + \
    df['Genres'] + ' ' + \
    df['Episodes'].astype(str) + ' ' + \
    df['AnimeStatus'] + ' ' + \
    df['Duration_min'].astype(str) + ' ' + \
    df['Popularity'].astype(str) + ' ' + \
    df['Rating'].astype(str)

# Tạo DataFrame mới chỉ gồm các cột cần thiết
new_df = df[['AnimeName', 'Description', 'Synopsis']]
```

- Sử dụng TfidfVectorizer để chuyển đổi văn bản thành dạng số (vector)

```
# Sử dụng TfidfVectorizer
tfidf_vec = TfidfVectorizer(stop_words='english', max_features=1500)
genre_vec = tfidf_vec.fit_transform(new_df['Description'])
```

- Hàm đề xuất bộ phim dựa vào cosine similarity

```
def get_recommend(title, cosine_sim, length=10):
    # Tạo một series để ánh xạ tên phim với chỉ số
    movie_title_series = pd.Series(df.index, index=new_df['AnimeName']).drop_duplicates()

    # Lấy chỉ số của bộ phim dựa trên tên
    if title not in movie_title_series:
        raise ValueError(f"'{title}' không tồn tại trong dữ liệu.")
    movie_idx = movie_title_series.get_loc(title)

    # Tính toán điểm tương đồng
    sim_scores = list(enumerate(cosine_sim[movie_idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:length+1] # Bỏ chính phim được chọn (phim đầu tiên)

    # Lấy tên phim và nội dung
    movie_indices = [i[0] for i in sim_scores]
    recommendations = new_df.iloc[movie_indices][['AnimeName', 'Synopsis']]
    return recommendations
```

- Đề xuất dựa trên danh sách người dùng đã chọn

```
def get_recommend_by_user_list(user_list, length=10):
    # Kiểm tra đầu vào
    if not user_list:
        raise ValueError("Danh sách phim người dùng không được để trống.")
    # Tạo vector người dùng bằng cách lấy trung bình vectơ của các phim
    user_vec = np.mean(
        [genre_vec[new_df[new_df['AnimeName'] == movie].index[0]].toarray()
         for movie in user_list if movie in new_df['AnimeName'].values],
        axis=0
    )
    # Tính toán độ tương đồng giữa vector người dùng và các bộ phim
    cosine = cosine_similarity(user_vec, genre_vec)
    # Sắp xếp các bộ phim theo độ tương đồng giảm dần
    sim_scores = list(enumerate(cosine[0]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # Loại bỏ các phim đã chọn trong danh sách người dùng
    sim_scores = [score for score in sim_scores if new_df.iloc[score[0]]['AnimeName'] not in user_list]
    sim_scores = sim_scores[:length] # Chỉ lấy số lượng phim cần để xuất
    # Lấy tên phim và nội dung
    movie_indices = [i[0] for i in sim_scores]
    recommendations = new_df.iloc[movie_indices][['AnimeName', 'Synopsis']]
    return recommendations
```

# R7: XÂY DỰNG HỆ THỐNG ĐỀ XUẤT PHIM



## THỰC HIỆN BẰNG THUẬT TOÁN

- Thực hiện đề xuất từ 1 bộ phim đã xem:

```
name = input('Nhập tên phim bạn đã coi: ')
# Tính độ tương đồng cosine
cos_similar = cosine_similarity(genre_vec, genre_vec)
# từ 1 phim để đề xuất ra
# Ví dụ sử dụng:
print(f'Kết quả đề xuất cho phim {name}:')
recommendation_1 = get_recommend(name, cos_similar, length=10)
display(recommendation_1)
```

- Kết quả:

Kết quả đề xuất cho phim Naruto:

	AnimeName	Synopsis
9738	Street Fighter II: Yomigaeru Fujiwara-Kyou - T...	An educational anime featuring Ryu, Ken, Chun...
5381	Naruto: Takigakure no Shitou - Ore ga Eiyuu Da...	After safely escorting the cowardly Takigakure...
5148	Yuu☆Yuu☆Hakusho (Movie)	Yusuke Urameshi is not exactly what you'd call...
2356	Boruto: Naruto the Movie	The spirited Boruto Uzumaki, son of Seventh Ho...
3362	Yuu☆Yuu☆Hakusho: Eizou Hakusho - Ankoku Bujuts...	These are two recap specials that focus on Tea...
2050	Cang Yuan Tu: Dongning Fu Fanwei Pian	No synopsis information has been added to this...
2157	Feng Ling Liu Xiu	It is rumored in the martial world that a myst...
9753	Ninkuu: Knife no Bohyou	After the war, Fuusuke, Aicho and Touji head t...
2456	Naruto x UT	All-new animation offered throughout UNIQLO cl...
3452	Yuu☆Yuu☆Hakusho: Meikai Shitou-hen - Honoo no ...	Millennia ago, a war was fought between the Ne...

- Thực hiện đề xuất cho danh sách các phim:

```
print("\nKết quả đề xuất dựa trên danh sách người dùng:")
user_selected_movies = ['Dragon Ball', 'Bleach', 'Boku no Hero Academia']
recommendations = get_recommend_by_user_list(user_selected_movies, length=10)
display(recommendations)
```

- Kết quả

Kết quả đề xuất dựa trên danh sách người dùng:

	AnimeName	Synopsis
2153	Boku no Hero Academia the Movie 3: World Hero...	Adaptation of "NoXXX Hawks: SOOTHE" side-chap...
2484	Boku no Hero Academia the Movie 1: Futari no H...	Episode 1 is based on one-shot manga released ...
2861	Boku no Hero Academia: Hero Note	Recap of Boku no Hero Academia that aired a we...
3472	Boku no Hero Academia: Ikinokore! Kesshi no Su...	The stage this time is a survival training cou...
6158	Boku no Hero Academia: Memories	Recap of Boku no Hero Academia leading up to t...
3148	Boku no Hero Academia: Training of the Dead	Returning from their internships, the students...
3147	Boku no Hero Academia: Sukuel Kyuujo Kunren!	UA High School must regain the public's confid...
2448	Kekkai Sensen: Ousama no Restaurant no Ousama	For a rare occasion, the members of Libra have...
3256	Hunter x Hunter Pilot	Gon Freecss, a young boy living on Whale Islan...
298	Boku no Hero Academia 6th Season	With Tomura Shigaraki at its helm, the former ...



## ĐƯA LÊN WEBSITE

- Template Login

- Template Recommendation

- Template Dashboard

## R7: XÂY DỰNG HỆ THỐNG ĐỀ XUẤT PHIM



### KẾT LUẬN

Với mong muốn mang lại trải nghiệm gợi ý phim tốt, phù hợp cho người dùng, nhóm 1 đã xây dựng thành công Hệ thống đề xuất Anime với:

#### - **Ưu điểm:**

- + **Khả năng linh hoạt cao:** Người dùng có thể tùy chỉnh số lượng phim muốn đề xuất (length)
- + **Đa dạng, tiện ích:** Người dùng nhận được đề xuất một hoặc nhiều bộ phim dựa trên danh sách phim yêu thích hoặc từ lịch sử xem phim của bản thân

#### - **Nhược điểm:**

- + **Phụ thuộc nhiều vào dữ liệu đầu vào:** Do giới hạn về đầu vào của dữ liệu, khó có thể đề xuất phim đang hot hoặc theo trend
- + **Hạn chế về việc “hiểu” được gu xem** của người dùng: Nếu như người dùng thích bộ Anime có nhiều thể loại thì hệ thống không hiểu được sự kết hợp này.
- + **Dễ bị lặp lại các đề xuất:** Nếu có nhiều bộ phim có nội dung giống nhau thì hệ thống sẽ lặp lại đề xuất tương tự làm cho việc đề xuất trở nên “nhàm chán”
- + **Không phân biệt được chất lượng phim:** Hệ thống không biết phim nào được đánh giá cao/ thấp mà chỉ gợi ý những bộ phim tương đồng với list phim.



### ĐỀ XUẤT CẢI TIẾN

- **Tăng tính cá nhân hóa:** Thêm thông tin như lịch sử xem, thời gian xem, hành vi click để hiểu rõ hơn sở thích của người dùng

- **Tự động cập nhật dữ liệu:** Tạo pipeline tự động để cập nhật anime mới và tính toán lại các đặc trưng khi có dữ liệu mới

- **Quan tâm tới giao diện người dùng:** Visualization các dữ liệu giúp người dùng dễ dàng tiếp cận và hiểu các đề xuất hơn



### QUÁ TRÌNH THỰC HIỆN

Dự án không chỉ giúp nhóm áp dụng được các kiến thức đã học vào thực hành mà còn là cơ hội để nhóm phát triển các kỹ năng: xử lý dữ liệu, phân tích hệ thống, khả năng làm việc nhóm,...

#### - **Thuận lợi:**

- + Thành viên nhóm hiểu rất rõ về chủ đề của nhóm, tạo động lực và hứng thú thực hiện dự án
- + Dữ liệu về Anime ở trên web MyAnimeList phong phú, dễ hiểu và sử dụng
- + Các thành viên trong nhóm đều có trách nhiệm, nhiệt tình đóng góp, xây dựng bài của nhóm, cùng nhau phát triển và học hỏi
- + Có các công cụ, thư viện hỗ trợ mạnh mẽ: nâng cao khả năng hoàn thành dự án, tăng hiệu quả của dự án.

#### - **Khó khăn:**

- + Thiết bị: laptop, máy tính,... đôi khi khó xử lý dữ liệu lớn: tiêu tốn nhiều thời gian và bộ nhớ
- + Bị giới hạn về thời gian để phát triển thêm dự án
- + Sự thay đổi trong yêu cầu của dự án