**TRƯỜNG ĐẠI HỌC QUỐC TẾ HỒNG BÀNG**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

**🙢🕮🙠**



**BÁO CÁO ĐỒ ÁN**

**PHƯƠNG PHÁP PHÂN TÍCH**

**DỮ LIỆU CƠ BẢN**

**Giảng viên hướng dẫn : Lê Văn Hạnh**

**Sinh viên thực hiện : Nguyễn Minh Vũ**

**Mã số sinh viên : 2211110063**

**TP. Hồ Chí Minh, 2024**

TRANG CAM KẾT

Tôi xin cam kết báo cáo thường kỳ này được hoàn thành dựa trên các kết quả thực hiện bài thực hành của tôi và các mã nguồn và kết quả này chưa được dùng cho bất cứ báo cáo của sinh viên nào khác.

*TP.HCM, ngày ….. tháng ….. năm …2024..*

Người thực hiện

Nguyễn Minh Vũ

NHẬN XÉT CỦA GIẢNG VIÊN

TP.HCM, Ngày Tháng Năm 2024

Chữ ký giảng viên

# GIỚI THIỆU VỀ BỘ DỮ LIỆU CẦN DÙNG

# Youtube Trending Videos Dataset

***Giới thiệu về tập dữ liệu***

Bộ dữ liệu này gồm 18 cột và 162 nghìn dòng cung cấp thông tin chi tiết về mức độ tương tác khác nhau như số lượt xem video đã nhận được, lượt thích và không thích mà video đã thu thập được từ lượng người xem. Ngoài ra, thông tin liên quan đến số lượng bình luận trên các video cụ thể cho phép phân tích về tương tác và phản hồi của người xem. Hơn nữa, tập dữ liệu này mô tả liệu bình luận hoặc xếp hạng có bị vô hiệu hóa đối với một video cụ thể hay không, cho phép kiểm tra cách các yếu tố này tác động đến mức độ tương tác.

Bằng cách khám phá tập dữ liệu này, các nhà tiếp thị chuyên sâu có thể có được những hiểu biết có giá trị trong việc xác định xu hướng về mức độ phổ biến của nội dung trên nhiều quốc gia khác nhau trong khi vẫn tính đến các cân nhắc về thời gian dựa trên ngày trong tuần được công bố. Nó cũng mở ra các con đường để phân tích tình cảm của công chúng đối với các video cụ thể dựa trên tỷ lệ thích so với không thích và số lượng bình luận, từ đó hỗ trợ thêm cho việc đưa ra các chiến lược tiếp thị phù hợp.

Nhìn chung, tập dữ liệu thông tin này đóng vai trò là tài sản vô giá đối với các nhà nghiên cứu, nhà phân tích dữ liệu và nhà tiếp thị, những người nỗ lực tìm hiểu sâu hơn về các mẫu video thịnh hành, các số liệu liên quan ảnh hưởng đến tính lan truyền của nội dung, các yếu tố quyết định cảm xúc của người xem và khám phá những khả năng mới trong không gian tiếp thị kỹ thuật số tận dụng phạm vi tiếp cận rộng rãi của YouTube.

***Nguồn tài liệu***

Kaggle: **YouTube Trending Videos Dataset**

<https://www.kaggle.com/datasets/thedevastator/youtube-trending-videos-dataset/data>

***Các tính năng chính của tập dữ liệu***

1. *index* (Số thứ tự, kiểu integer): Thứ tự của các videos trending bắt đầu từ 0.
2. *video\_id* (ID Videos, kiểu string): Mã định danh duy nhất cho mỗi video.
3. *trending*\_date (kiểu date): Ngày mà video trở thành xu hướng.
4. *title* (kiểu text): Tựa đề của video cho người đăng tải chủ động đặt.
5. *channel\_title* (kiểu text): Tiêu đề của Youtube mà người phát hành đã thành lập.
6. *category\_id* (kiểu integer): Mã loại video được đặt bởi những chuyên gia.

|  |  |
| --- | --- |
| 1 | Film & Animation |
| 2 | Autos & Vehicles |
| 10 | Music |
| 15 | Pets & Animals |
| 17 | Sports |
| 18 | Short Movies |
| 19 | Travel & Events |
| 20 | Gaming |
| 21 | Videoblogging |
| 22 | People & Blogs |
| 23 | Comedy |
| 24 | Entertainment |
| 25 | News & Politics |
| 26 | Howto & Style |
| 27 | Education |
| 28 | Science & Technology |
| 29 | Nonprofits & Activism |
| 30 | Movies |
| 31 | Anime/Animation |
| 32 | Action/Adventure |
| 33 | Classics |
| 34 | Comedy |
| 35 | Documentary |
| 36 | Drama |
| 37 | Family |
| 38 | Foreign |
| 39 | Horror |
| 40 | Sci-Fi/Fantasy |
| 41 | Thriller |
| 42 | Shorts |
| 43 | Shows |
| 44 | Trailers |

1. *publish\_date* (kiểu date): Ngày mà video được đăng tải trên Youtube.
2. *time\_frame* (kiểu text, \*time to time): Khoảng thời gian video trở nên xu hướng.
3. *published\_day\_of\_week* (Ngày trong tuần mà video xu hướng, thuộc tính dạng danh nghĩa - categorical): gồm Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.
4. *publish\_country* (Quốc gia, thuộc tính dạng danh nghĩa - categorical): gồm US, CANADA, FRANCE, GB.
5. *tags* (kiểu text): Các thẻ hoặc từ khóa liên quan đến video.
6. *views* (kiểu integer): Số lượng người xem mà video đạt được.
7. *likes* (kiểu integer): Số lượng người thích video.
8. *dislikes* (kiểu integer): Số lượng người không thích video.
9. *comment\_count* (kiểu integer): Số lượng bình luận của video.
10. comments\_disabled *comments\_disabled* (thuộc tính dạng boolean): Chỉ ra liệu bình luận có bị vô hiệu hóa cho video hay không (True/False).
11. ratings\_disabled *ratings\_disabled* (thuộc tính dạng boolean): Cho biết liệu xếp hạng có bị vô hiệu hóa đối với video hay không (True/False).
12. *video\_error\_or\_removed* (thuộc tính dạng boolean): Cho biết liệu video đó có bị lỗi hoặc đã bị xóa hay không (True/False).

# PHIẾU KHẢO SÁT MINH HỌA CHI TIẾT

Trending date: Click or tap to enter a date.Choose Date

Title: 

Channel title: 

Category ID: Choose ID

Publish date: Choose Date

Time frame: Choose an item.Choose Hour : Choose Minute to Choose Hour Choose an item.: Choose Minute

Publish day of week: Choose day

Publish country:    

Views:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (number)

Likes:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (number)

Dislikes:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (number)

Comment count: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (number)

Comment disable: 

Ratings disable: 

Video error or removed: 

# QUÁ TRÌNH TIỀN XỬ LÝ DỮ LIỆU - SQL

## Làm sạch dữ liệu

### Điền giá trị còn thiếu

* Chủ động xóa các fields

Nếu dữ liệu không có giá trị NULL, SV chủ động xóa trên tất cả các field (ngoại trừ field ID), mỗi field khoảng 5 giá trị (trên 5 record liền nhau để dễ quan sát kết quả).

***Trending\_date***

UPDATE Youtube\_Test

SET trending\_date = '20177-11-14'

WHERE Youtube\_Test.[index] BETWEEN 10 AND 14

***Publish\_date***

UPDATE Youtube\_Test

SET publish\_date = '17-11-13'

WHERE Youtube\_Test.[index] BETWEEN 5 AND 9

***Views***

UPDATE Youtube\_Table

SET views = NULL

WHERE Youtube\_Table.[index] BETWEEN 25 AND 29

***Likes***

UPDATE Youtube\_Table

SET likes = NULL

WHERE Youtube\_Table.[index] BETWEEN 30 AND 34

***Dislikes***

UPDATE Youtube\_Table

SET dislikes = NULL

WHERE Youtube\_Table.[index] BETWEEN 35 AND 39

***Comment\_count***

UPDATE Youtube\_Table

SET comment\_count = NULL

WHERE Youtube\_Table.[index] BETWEEN 40 AND 44

***Comments\_disabled***

UPDATE Youtube\_Table

SET comments\_disabled = NULL

WHERE Youtube\_Table.[index] BETWEEN 45 AND 49

***Ratings\_disabled***

UPDATE Youtube\_Table

SET ratings\_disabled = NULL

WHERE Youtube\_Table.[index] BETWEEN 50 AND 54

***Video\_error\_or\_removed***

UPDATE Youtube\_Table

SET video\_error\_or\_removed = NULL

WHERE Youtube\_Table.[index] BETWEEN 55 AND 59

***Tags***

UPDATE Youtube\_Table

SET tags = NULL

WHERE Youtube\_Table.[index] BETWEEN 65 AND 69

A screen shot of a computer program

Description automatically generated

* Điền các giá trị vào fields NULL

Không điền những giá trị đơn giản (như “không biết”, “không có”, hay min, max, mean, mode, median) cho những giá trị NULL mà cần điền bằng những giá trị đòi hỏi tính toán phức tạp hơn, ví dụ điền giá trị NULL bằng giá trị của mode, nếu có nhiều mode thì chọn mode có giá trị thấp nhất hoặc đối với kiểu số thực, có thể thực hiện làm tròn (đến phần nguyên hoặc phần ngàn, …) trước khi tính mean/mode/median.

***Views***

WITH CTE

AS

( --mean

SELECT CAST(AVG(CAST(views AS BIGINT)) AS BIGINT) AS AvgViews

FROM Youtube\_Test

WHERE views IS NOT NULL

UNION

--mode

SELECT TOP 1 views

FROM Youtube\_Test

WHERE views IS NOT NULL

GROUP BY views

ORDER BY COUNT(\*) DESC

UNION

--median

SELECT AVG(views)

FROM Youtube\_Test

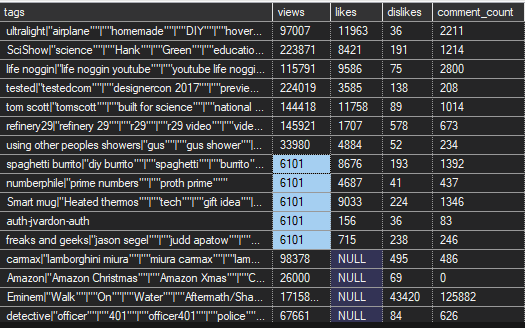
WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test) AND views IS NOT NULL

)

UPDATE Youtube\_Test

SET views = (SELECT MIN(AvgViews) FROM CTE)

WHERE views IS NULL



***Likes***

WITH CTE

AS

(

--mean

SELECT CAST(AVG(CAST(likes AS BIGINT)) AS BIGINT) AS AvgLikes

FROM Youtube\_Test

WHERE likes IS NOT NULL

UNION

--mode

SELECT TOP 1 likes

FROM Youtube\_Test

WHERE likes IS NOT NULL

GROUP BY likes

ORDER BY COUNT(likes) DESC

UNION

--median

SELECT AVG(likes)

FROM Youtube\_Test

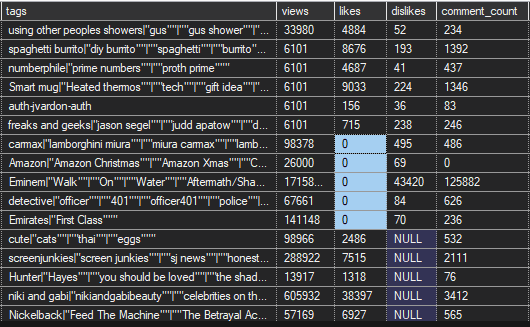
WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test) AND likes IS NOT NULL

)

UPDATE Youtube\_Test

SET likes = (SELECT MIN(AvgLikes) FROM CTE)

WHERE likes IS NULL



***Dislikes***

WITH CTE

AS

(

--mean

SELECT CAST(AVG(CAST(dislikes AS BIGINT)) AS BIGINT) AS AvgDislikes

FROM Youtube\_Test

WHERE dislikes IS NOT NULL

UNION

--mode

SELECT TOP 1 dislikes

FROM Youtube\_Test

WHERE dislikes IS NOT NULL

GROUP BY dislikes

ORDER BY (COUNT(dislikes)) DESC

UNION

--median

SELECT AVG(dislikes)

FROM Youtube\_Test

WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test) AND dislikes IS NOT NULL

)

UPDATE Youtube\_Test

SET dislikes = (SELECT MIN(Avgdislikes) FROM CTE)

WHERE dislikes IS NULL

A screenshot of a black screen

Description automatically generated

***Comment\_count***

WITH CTE

AS

(

--mean

SELECT CAST(AVG(CAST(comment\_count AS BIGINT)) AS BIGINT) AS AvgCommentcount

FROM Youtube\_Test

WHERE comment\_count IS NOT NULL

UNION

--mode

SELECT TOP 1 comment\_count

FROM Youtube\_Test

WHERE comment\_count IS NOT NULL

GROUP BY comment\_count

ORDER BY (COUNT(comment\_count)) DESC

UNION

--median

SELECT AVG(comment\_count)

FROM Youtube\_Test

WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test) AND comment\_count IS NOT NULL

)

UPDATE Youtube\_Test

SET comment\_count = (SELECT MIN(AvgCommentcount) FROM CTE)

WHERE comment\_count IS NULL

SELECT \* FROM Youtube\_Table\_2

ORDER BY Youtube\_Table\_2.[index] ASC

A black and white screen with numbers and text

Description automatically generated

***Comments\_disabled***

--Update theo xac suat

DECLARE @PercentageZero FLOAT;

SET @PercentageZero = (SELECT CAST(COUNT(\*) AS FLOAT) / ((SELECT COUNT(\*) FROM Youtube\_Test) - (SELECT COUNT(\*) FROM Youtube\_Test WHERE comments\_disabled IS NULL))

FROM Youtube\_Test

WHERE comments\_disabled = '0');

DECLARE @Index INT;

DECLARE @comments\_disabled INT;

DECLARE YT\_cursor CURSOR FOR

SELECT [index] FROM Youtube\_Test WHERE comments\_disabled IS NULL;

OPEN YT\_cursor;

FETCH NEXT FROM YT\_cursor INTO @Index;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF RAND(CHECKSUM(NEWID())) <= @PercentageZero

BEGIN

SET @comments\_disabled = 0;

END

ELSE

BEGIN

SET @comments\_disabled = 1;

END

UPDATE Youtube\_Test

SET comments\_disabled = @comments\_disabled

WHERE [index] = @Index;

FETCH NEXT FROM YT\_cursor INTO @Index;

END;

CLOSE YT\_cursor;

DEALLOCATE YT\_cursor;

***Ratings\_disabled***

--Ratings Disabled

DECLARE @PercentageZero FLOAT;

SET @PercentageZero = (SELECT CAST(COUNT(\*) AS FLOAT) / ((SELECT COUNT(\*) FROM Youtube\_Test) - (SELECT COUNT(\*) FROM Youtube\_Test WHERE ratings\_disabled IS NULL))

FROM Youtube\_Test

WHERE ratings\_disabled = '0');

DECLARE @Index INT;

DECLARE @ratings\_disabled INT;

DECLARE YT\_cursor CURSOR FOR

SELECT [index] FROM Youtube\_Test WHERE ratings\_disabled IS NULL;

OPEN YT\_cursor;

FETCH NEXT FROM YT\_cursor INTO @Index;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF RAND(CHECKSUM(NEWID())) <= @PercentageZero

BEGIN

SET @ratings\_disabled = 0;

END

ELSE

BEGIN

SET @ratings\_disabled = 1;

END

UPDATE Youtube\_Test

SET ratings\_disabled = @ratings\_disabled

WHERE [index] = @Index;

FETCH NEXT FROM YT\_cursor INTO @Index;

END;

CLOSE YT\_cursor;

DEALLOCATE YT\_cursor;

***Video\_error\_or\_removed***

DECLARE @PercentageZero FLOAT;

SET @PercentageZero = (SELECT CAST(COUNT(\*) AS FLOAT) / ((SELECT COUNT(\*) FROM Youtube\_Test) - (SELECT COUNT(\*) FROM Youtube\_Test WHERE video\_error\_or\_removed IS NULL))

FROM Youtube\_Test

WHERE video\_error\_or\_removed = '0');

DECLARE @Index INT;

DECLARE @video\_error\_or\_removed INT;

DECLARE YT\_cursor CURSOR FOR

SELECT [index] FROM Youtube\_Test WHERE video\_error\_or\_removed IS NULL;

OPEN YT\_cursor;

FETCH NEXT FROM YT\_cursor INTO @Index;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF RAND(CHECKSUM(NEWID())) <= @PercentageZero

BEGIN

SET @video\_error\_or\_removed = 0;

END

ELSE

BEGIN

SET @video\_error\_or\_removed = 1;

END

UPDATE Youtube\_Test

SET video\_error\_or\_removed = @video\_error\_or\_removed

WHERE [index] = @Index;

FETCH NEXT FROM YT\_cursor INTO @Index;

END;

CLOSE YT\_cursor;

DEALLOCATE YT\_cursor;

SELECT \* FROM Youtube\_Test

ORDER BY Youtube\_Test.[index] ASC

A screenshot of a computer

Description automatically generated

***Tags***

UPDATE Youtube\_Test

SET tags = (SELECT CONCAT(SUBSTRING(title, 1, 6), '|"', SUBSTRING(title, 9, 6), '""|""', SUBSTRING(title, 19, 7), '""|""', SUBSTRING(title, 30, 17), '""|""', '2018', '""|""', 'Movie', '""|""', (SELECT channel\_title FROM Youtube\_Test WHERE [index] = 65), '"""')

FROM Youtube\_Test

WHERE [index] = 65)

WHERE [index] = 65

UPDATE Youtube\_Test

SET tags = (SELECT CONCAT(SUBSTRING(title, 1, 7), '|"', SUBSTRING(title, 9, 21), '""|""', SUBSTRING(title, 23, 12), '""|""', SUBSTRING(title, 40, 6), '""|""', (SELECT channel\_title FROM Youtube\_Test WHERE [index] = 66), '"""')

FROM Youtube\_Test

WHERE [index] = 66)

WHERE [index] = 66

UPDATE Youtube\_Test

SET tags = (SELECT CONCAT(SUBSTRING(title, 1, 7), '|"', SUBSTRING(title, 9, 19), '""|""', SUBSTRING(title, 12, 6), '""|""', SUBSTRING(title, 12, 16), '""|""', SUBSTRING(title, 21, 7), '""|""', (SELECT channel\_title FROM Youtube\_Test WHERE [index] = 67), '"""')

FROM Youtube\_Test

WHERE [index] = 67)

WHERE [index] = 67

UPDATE Youtube\_Test

SET tags = (SELECT CONCAT(SUBSTRING(title, 1, 5), '|"', SUBSTRING(title, 7, 6), '""|""', SUBSTRING(title, 14, 6), '""|""', SUBSTRING(title, 7, 13), '""|""', SUBSTRING(title, 26, 36), '""|""', SUBSTRING(title, 49, 12), '""|""', SUBSTRING(title, 63, 6), '""|""', (SELECT channel\_title FROM Youtube\_Test WHERE [index] = 68), '"""')

FROM Youtube\_Test

WHERE [index] = 68)

WHERE [index] = 68

UPDATE Youtube\_Test

SET tags = (SELECT CONCAT(SUBSTRING(title, 1, 10), '|"', SUBSTRING(title, 12, 9), '""|""', SUBSTRING(title, 25, 4), '""|""', SUBSTRING(title, 33, 14), '"""')

FROM Youtube\_Test

WHERE [index] = 69)

WHERE [index] = 69

A screenshot of a computer screen

Description automatically generated

### Làm mịn dữ liệu

Chọn thuộc tính có miền giá trị cao nhất trong dữ liệu. Thực hiện chia miền giá trị này

thành nhiều nhóm (>=5 nhóm). Chọn 1 trong 3 cách sau để làm mịn dữ liệu:

 Thay thế bằng giá trị trung bình (mean) của nhóm.

 Thay thế bằng giá trị trung vị (median) của nhóm.

 Làm mịn theo ranh giới của nhóm (smoothing by bin boundaries)

* Chia thành 10 nhóm và theo 3 cách khác nhau
* Mean

WITH RankedData

AS

(

SELECT views, NTILE(10) OVER (ORDER BY views ASC) AS range\_group

FROM Youtube\_Test

),

RangeAverages

AS

(

SELECT range\_group, CAST(AVG(CAST(views AS BIGINT)) AS BIGINT) AS avg\_views

FROM RankedData

GROUP BY range\_group

)

UPDATE yt

SET yt.views = ra.avg\_views

FROM Youtube\_Test yt

JOIN RankedData rd ON yt.views = rd.views

JOIN RangeAverages ra ON rd.range\_group = ra.range\_group;

* Median

SELECT views INTO view\_median

FROM Youtube\_Table

WITH RankedData2

AS

(

SELECT views, NTILE(10) OVER (ORDER BY views ASC) AS range\_group

FROM view\_median

),

RangeMedian

AS

(

SELECT range\_group, (CAST(MIN(views) AS BIGINT) + CAST(MAX(views) AS BIGINT)) / 2 AS median\_views

FROM RankedData2

GROUP BY range\_group

)

UPDATE yt

SET yt.views = rm.median\_views

FROM view\_median yt

JOIN RankedData2 rd ON yt.views = rd.views

JOIN RangeMedian rm ON rd.range\_group = rm.range\_group;

SELECT \* FROM view\_median ORDER BY views

* Boundaries

SELECT views INTO view\_boundaries

FROM Youtube\_Table

WITH RankedData3 AS (

SELECT

views,

NTILE(10) OVER (ORDER BY views ASC) AS range\_group

FROM view\_boundaries

),

RangeBoundaries AS (

SELECT

range\_group,

CASE

WHEN ABS(CAST(MAX(views) AS BIGINT) - MAX(views)) > ABS(MAX(views) - CAST(MIN(views) AS BIGINT))

THEN CAST(MIN(views) AS BIGINT)

ELSE CAST(MAX(views) AS BIGINT)

END AS boundaries\_views

FROM RankedData3

GROUP BY range\_group

)

UPDATE yt

SET yt.views = rb.boundaries\_views

FROM view\_boundaries yt

JOIN RankedData3 rd ON yt.views = rd.views

JOIN RangeBoundaries rb ON rd.range\_group = rb.range\_group;

SELECT \* FROM view\_boundaries ORDER BY views

Chọn field có độ lệch chuẩn lớn nhất trong dữ liệu: thực hiện tăng 10% giá trị của thuộc tính này cho 5 giá trị nhỏ nhất và giảm 10% giá trị của thuộc tính này cho 5 giá trị lớn nhất.

SELECT \* INTO Youtube\_Table\_2 FROM Youtube\_Table

SELECT \* FROM Youtube\_Table\_2

WITH CTE1 AS

(

SELECT ROW\_NUMBER() OVER (ORDER BY ORDINAL\_POSITION) AS ColIndex,

COLUMN\_NAME AS ColName

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = 'Youtube\_Test' AND ORDINAL\_POSITION BETWEEN 12 AND 15

),

CTE2 AS

(

SELECT 1 AS ColIndex, CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(views AS FLOAT), 2)) - POWER(AVG(CAST(views AS FLOAT)), 2)) AS BIGINT) AS dlc

FROM Youtube\_Test

UNION ALL

SELECT 2 AS ColIndex, CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(likes AS FLOAT), 2)) - POWER(AVG(CAST(likes AS FLOAT)), 2)) AS BIGINT)

FROM Youtube\_Test

UNION ALL

SELECT 3 AS ColIndex, CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(dislikes AS FLOAT), 2)) - POWER(AVG(CAST(dislikes AS FLOAT)), 2)) AS BIGINT)

FROM Youtube\_Test

UNION ALL

SELECT 4 AS ColIndex, CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(comment\_count AS FLOAT), 2)) - POWER(AVG(CAST(comment\_count AS FLOAT)), 2)) AS BIGINT)

FROM Youtube\_Test

)

SELECT c1.ColName, c2.dlc

FROM CTE1 c1

JOIN CTE2 c2 ON c1.ColIndex = c2.ColIndex;

--Tim va thay doi 5 giá trị nhỏ nhất trong views

UPDATE Youtube\_Test

SET views = views \* 1.1

WHERE [index] IN (SELECT TOP 5 [index] FROM Youtube\_Test ORDER BY views ASC)

--Tìm va thay doi 5 giá trị lớn nhất trong views

UPDATE Youtube\_Test

SET views = views \* 0.9

WHERE [index] IN (SELECT TOP 5 [index] FROM Youtube\_Test ORDER BY views DESC)

A screenshot of a computer

Description automatically generated

### Xác định giá trị ngoại lệ

Xác định dữ liệu có chứa giá trị ngoại lệ (tính theo five number summary) của từng thuộc tính số hay không?

--Views

WITH CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) views AS views\_q1

FROM Youtube\_Test

ORDER BY views ASC

), CTE2

AS

(

--Q2

SELECT TOP ( SELECT COUNT(\*) / 2 FROM Youtube\_Test) views AS views\_q2

FROM Youtube\_Test

ORDER BY views ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) views AS views\_q3

FROM Youtube\_Test

ORDER BY views ASC

)

SELECT \*

FROM Youtube\_Test

WHERE views < ((SELECT MAX(views\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1))) OR views > ((SELECT MAX(views\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1)))

ORDER BY views ASC

A screen shot of a computer

Description automatically generated

--Likes

WITH CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) likes AS likes\_q1

FROM Youtube\_Test

ORDER BY likes ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) likes AS likes\_q3

FROM Youtube\_Test

ORDER BY likes ASC

)

SELECT \*

FROM Youtube\_Test

WHERE likes < ((SELECT MAX(likes\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(likes\_q3) FROM CTE3) - (SELECT MAX(likes\_q1) FROM CTE1))) OR likes > ((SELECT MAX(likes\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(likes\_q3) FROM CTE3) - (SELECT MAX(likes\_q1) FROM CTE1)))

ORDER BY likes ASC

A screen shot of a computer program

Description automatically generated

--Dislikes

WITH CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) dislikes AS dislikes\_q1

FROM Youtube\_Test

ORDER BY dislikes ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) dislikes AS dislikes\_q3

FROM Youtube\_Test

ORDER BY dislikes ASC

)

SELECT \*

FROM Youtube\_Test

WHERE dislikes < ((SELECT MAX(dislikes\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(dislikes\_q3) FROM CTE3) - (SELECT MAX(dislikes\_q1) FROM CTE1))) OR dislikes > ((SELECT MAX(dislikes\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(dislikes\_q3) FROM CTE3) - (SELECT MAX(dislikes\_q1) FROM CTE1)))

ORDER BY dislikes ASC

A screen shot of a computer program

Description automatically generated

--Comment Count

WITH CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) comment\_count AS comment\_q1

FROM Youtube\_Test

ORDER BY comment\_count ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) comment\_count AS comment\_q3

FROM Youtube\_Test

ORDER BY comment\_count ASC

)

SELECT \*

FROM Youtube\_Test

WHERE comment\_count < ((SELECT MAX(comment\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(comment\_q3) FROM CTE3) - (SELECT MAX(comment\_q1) FROM CTE1))) OR comment\_count > ((SELECT MAX(comment\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(comment\_q3) FROM CTE3) - (SELECT MAX(comment\_q1) FROM CTE1)))

ORDER BY comment\_count ASC

A screen shot of a computer screen

Description automatically generated

### Xử lý dữ liệu không nhất quán

* Đối với field có kiểu dữ liệu ngày: từ file dữ liệu gốc, SV thực hiện đổi kiểu dữ liệu thành chuỗi, sau đó sửa thủ công các giá trị ngày cho ít nhất 5 giá trị để có 5 dữ liệu có định dạng ngày khác với các record còn lại. Ví dụ định dạng của ngày hiện tại là mm/dd/yy, sửa 5 giá trị để 5 giá trị này có kiểu cùng là dd/mm/yy hoặc cùng có kiểu là yy/mm/dd. Sau đó thực hiện cập nhật lại giá trị cho 5 giá trị này.
* Đối với field kiểu danh nghĩa, SV cũng chủ động chỉnh sửa 5 giá trị, sau đó dủng lệnh thực hiện hiệu chỉnh lại cho đúng với giá trị gốc

***Trending\_date, Publish\_date***

A screenshot of a computer

Description automatically generated

SELECT \* FROM Youtube\_Test

ORDER BY Youtube\_Test.[index] ASC

--Convert Date to String

SELECT CONVERT(NVARCHAR, trending\_date, 120) AS ConvertedString

FROM Youtube\_Table;

SELECT CONVERT(NVARCHAR, publish\_date, 120) AS ConvertedString

FROM Youtube\_Table;

ALTER TABLE Youtube\_Test

ALTER COLUMN trending\_date NVARCHAR(MAX);

--Kiem tra kieu du lieu

SELECT COLUMN\_NAME, DATA\_TYPE, CHARACTER\_MAXIMUM\_LENGTH

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = 'Youtube\_Table' AND COLUMN\_NAME = 'trending\_date';

--20177-11-14 -> 2017-11-14

SELECT trending\_date

FROM Youtube\_Test

WHERE trending\_date NOT LIKE '%-\_\_-%'

UPDATE a

SET a.trending\_date = REPLACE(a.trending\_date, SUBSTRING(a.trending\_date, 1, 5), '2017')

FROM Youtube\_Test a

WHERE a.trending\_date NOT LIKE '\_\_\_\_-%';

--2017-111-14 -> 2017-11-14

UPDATE a

SET a.trending\_date = REPLACE(a.trending\_date, SUBSTRING(a.trending\_date, 6, 3), '11')

FROM Youtube\_Test a

WHERE a.trending\_date NOT LIKE '%-\_\_-%';

--2017-11-144 -> 2017-11-14

UPDATE a

SET a.trending\_date = REPLACE(a.trending\_date, SUBSTRING(a.trending\_date, 9, 3), '14')

FROM Youtube\_Test a

WHERE a.trending\_date NOT LIKE '%-\_\_';

--2017-13-11 -> 2017-11-13

UPDATE a

SET a.publish\_date = ( SELECT SUBSTRING(CONCAT(publish\_date, '-' + SUBSTRING(publish\_date, 6, 2)), 1, 4) + SUBSTRING(CONCAT(publish\_date, '-' + SUBSTRING(publish\_date, 6, 2)), 8, 6)

FROM Youtube\_Test

WHERE publish\_date LIKE '\_\_\_\_-[1][3-9]-\_\_' AND a.[index] = Youtube\_Test.[index])

FROM Youtube\_Test a

WHERE publish\_date LIKE '\_\_\_\_-[1][3-9]-\_\_';

--11-13-2017 -> 2017-11-13

UPDATE a

SET a.publish\_date = ( SELECT SUBSTRING(CONCAT(publish\_date, '-' + SUBSTRING(publish\_date, 1, 2) + '-' + SUBSTRING(publish\_date, 4, 2)), 7, 17)

FROM Youtube\_Test

WHERE publish\_date LIKE '\_\_-[1][3-9]-\_\_\_\_' AND a.[index] = Youtube\_Test.[index])

FROM Youtube\_Test a

WHERE publish\_date LIKE '\_\_-[1][3-9]-\_\_\_\_';

--13-11-2017 -> 2017-11-13

UPDATE a

SET a.publish\_date = ( SELECT SUBSTRING(CONCAT(publish\_date, '-' + SUBSTRING(publish\_date, 4, 2) + '-' + SUBSTRING(publish\_date, 1, 2)), 7, 17)

FROM Youtube\_Test

WHERE publish\_date LIKE '%-2017' AND a.[index] = Youtube\_Test.[index])

FROM Youtube\_Test a

WHERE publish\_date LIKE '%-2017';

--17-11-13 -> 2017-11-13

UPDATE a

SET a.publish\_date = ( SELECT SUBSTRING(CONCAT(publish\_date, '20' + SUBSTRING(publish\_date, 1, 2) + '-' + SUBSTRING(publish\_date, 4, 2)) + '-' + SUBSTRING(publish\_date, 7, 2), 9, 19)

FROM Youtube\_Test

WHERE publish\_date LIKE '\_\_-\_\_-\_\_' AND a.[index] = Youtube\_Test.[index])

FROM Youtube\_Test a

WHERE publish\_date LIKE '\_\_-\_\_-\_\_';

ALTER TABLE Youtube\_Test

ALTER COLUMN trending\_date DATE;

A screenshot of a computer

Description automatically generated

## Tích hợp dữ liệu

## Giảm thiểu dữ liệu

* Phân cụm

--Phan cum dua theo the loai videos va views(category\_id, views)

WITH CTE AS (

SELECT category\_id, SUM(CAST(views AS BIGINT)) AS Views\_PC

FROM Youtube\_Test

GROUP BY category\_id

)

SELECT category\_id,

Views\_PC,

CASE

WHEN Views\_PC >= 100000000 THEN 'High Value'

WHEN Views\_PC >= 1000000 THEN 'Medium Value'

ELSE 'Low Value'

END AS ValueGroup

FROM CTE;

A black and white table with numbers and a white text

Description automatically generated

* Chọn lựa mẫu

--Chon lua mau

SELECT \*

FROM Youtube\_Test

ORDER BY NEWID()

LIMIT 1000;

A screen shot of a computer

Description automatically generated

* Tổng hợp

--Tong hop du lieu views theo published\_day\_of\_week

SELECT published\_day\_of\_week, SUM(CAST(views AS BIGINT)) AS total\_views

FROM Youtube\_Test

GROUP BY published\_day\_of\_week;

A screenshot of a computer

Description automatically generated

## Chuyển đổi và phân tách dữ liệu

- Kiểu danh mục (ví dụ như các mùa: Xuân, Hạ, Thu, Đông): chuyển thành kiểu số nguyên.

- Kiểu thứ tự (ví dụ như: Giỏi, khá, trung bình, yếu): chuyển thành kiểu số nguyên. - Kiểu boolean: khi dữ liệu có từ 5 field kiểu này trở lên, chuyển 5 field thành 1 field kiểu số.

- Kiểu chuỗi: trong field kiểu chuỗi lại chứa danh sách các chuỗi con có ý nghĩa (ví dụ Tên hàng Đã mua trong đó gồm tên các sản phẩm đã mua trong đơn hàng như: bột giặt, khăn tắm, chén, khăn tắm): cần tách tên các sản phẩm ra khỏi field này để có nhiều field mới. Sau đó, do có thể có rất nhiều field mới nên cần chuyển đổi tất cả các field mới này trở lại thành 1 field duy nhất có kiểu là int.

Trong quá trình chuyển đổi có yêu cầu tạo ra các table làm trung gian, SV cần tìm cách gộp các table vừa phát sinh có cùng cấu trúc, ý nghĩa để giảm bớt số lượng table thực tế sẽ dùng.

***publish\_country***

CREATE TABLE Country\_Table (

country\_name NVARCHAR(20),

id INT IDENTITY(1,1)

)

INSERT INTO Country\_Table(country\_name)

SELECT DISTINCT publish\_country

FROM Youtube\_Test

ORDER BY publish\_country ASC

UPDATE Youtube\_Test

SET publish\_country = (SELECT id FROM Country\_Table WHERE Country\_Table.country\_name = Youtube\_Test.publish\_country)

WHERE publish\_country IN (SELECT country\_name FROM Country\_Table)

ALTER TABLE Youtube\_Test

ALTER COLUMN publish\_country INT;

SELECT \*

FROM Youtube\_Test

ORDER BY [index] ASC

A screenshot of a black and white screen

Description automatically generated

***published\_day\_of\_week***

CREATE TABLE DOW\_Table (

day\_name NVARCHAR(20),

id INT

)

INSERT INTO DOW\_Table (id, day\_name)

SELECT DISTINCT IIF(published\_day\_of\_week = 'Monday', 2, IIF(published\_day\_of\_week = 'Tuesday', 3, IIF(published\_day\_of\_week = 'Wednesday', 4, IIF(published\_day\_of\_week = 'Thursday', 5, IIF(published\_day\_of\_week = 'Friday', 6, IIF(published\_day\_of\_week = 'Saturday', 7, 1)))))) AS num, published\_day\_of\_week

FROM Youtube\_Test

WHERE published\_day\_of\_week NOT IN (SELECT day\_name FROM DOW\_Table)

SELECT \*

FROM DOW\_Table

UPDATE Youtube\_Test

SET published\_day\_of\_week = (SELECT id FROM DOW\_Table WHERE DOW\_Table.day\_name = Youtube\_Test.published\_day\_of\_week)

WHERE published\_day\_of\_week IN (SELECT day\_name FROM DOW\_Table)

ALTER TABLE Youtube\_Test

ALTER COLUMN published\_day\_of\_week INT;

A screenshot of a black screen

Description automatically generated

***time\_frame***

CREATE TABLE TF\_Table

(

id INT,

time\_frame NVARCHAR(200),

)

INSERT INTO TF\_Table (id, time\_frame)

SELECT DISTINCT CASE

WHEN time\_frame = '0:00 to 0:59' THEN 1

WHEN time\_frame = '1:00 to 1:59' THEN 2

WHEN time\_frame = '2:00 to 2:59' THEN 3

WHEN time\_frame = '3:00 to 3:59' THEN 4

WHEN time\_frame = '4:00 to 4:59' THEN 5

WHEN time\_frame = '5:00 to 5:59' THEN 6

WHEN time\_frame = '6:00 to 6:59' THEN 7

WHEN time\_frame = '7:00 to 7:59' THEN 8

WHEN time\_frame = '8:00 to 8:59' THEN 9

WHEN time\_frame = '9:00 to 9:59' THEN 10

WHEN time\_frame = '10:00 to 10:59' THEN 11

WHEN time\_frame = '11:00 to 11:59' THEN 12

WHEN time\_frame = '12:00 to 12:59' THEN 13

WHEN time\_frame = '13:00 to 13:59' THEN 14

WHEN time\_frame = '14:00 to 14:59' THEN 15

WHEN time\_frame = '15:00 to 15:59' THEN 16

WHEN time\_frame = '16:00 to 16:59' THEN 17

WHEN time\_frame = '17:00 to 17:59' THEN 18

WHEN time\_frame = '18:00 to 18:59' THEN 19

WHEN time\_frame = '19:00 to 19:59' THEN 20

WHEN time\_frame = '20:00 to 20:59' THEN 21

WHEN time\_frame = '21:00 to 21:59' THEN 22

WHEN time\_frame = '22:00 to 22:59' THEN 23

ELSE 24

END AS idtime, time\_frame

FROM Youtube\_Test

WHERE time\_frame NOT IN (SELECT time\_frame FROM TF\_Table)

SELECT DISTINCT time\_frame

FROM Youtube\_Test

UPDATE Youtube\_Test

SET time\_frame = (SELECT id FROM TF\_Table WHERE TF\_Table.time\_frame = Youtube\_Test.time\_frame)

WHERE time\_frame IN (SELECT time\_frame FROM TF\_Table)

ALTER TABLE Youtube\_Test

ALTER COLUMN time\_frame INT;

SELECT \*

FROM Youtube\_Test

ORDER BY [index] ASC

A screen shot of a computer

Description automatically generated

***Kieu boolean***

ALTER TABLE Youtube\_Test

ADD field\_bool INT

UPDATE m

SET m.field\_bool = (SELECT CONCAT(CAST(comments\_disabled AS NVARCHAR(1)), CAST(ratings\_disabled AS NVARCHAR(1)), CAST(video\_error\_or\_removed AS NVARCHAR(1))) FROM Youtube\_Test WHERE Youtube\_Test.[index] = m.[index])

FROM Youtube\_Test m;

CREATE TABLE CRVF (

comments\_disabled BIT,

ratings\_disabled BIT,

video\_error\_or\_removed BIT,

field\_bool INT

)

INSERT INTO CRVF(comments\_disabled, ratings\_disabled, video\_error\_or\_removed, field\_bool)

SELECT DISTINCT comments\_disabled, ratings\_disabled, video\_error\_or\_removed, field\_bool

FROM Youtube\_Test

ALTER TABLE Youtube\_Test

DROP COLUMN comments\_disabled, ratings\_disabled, video\_error\_or\_removed

SELECT DISTINCT trending\_date, publish\_date

FROM Youtube\_Test

ORDER BY trending\_date

A screenshot of a computer

Description automatically generated

***Trong quá trình chuyển đổi có yêu cầu tạo ra các table làm trung gian, SV cần tìm cách gộp các table vừa phát sinh có cùng cấu trúc, ý nghĩa để giảm bớt số lượng table thực tế sẽ dùng.***

CREATE TABLE Total\_Table (

col1 INT,

col2 BIT,

col3 BIT,

col4 BIT,

col5 INT,

col6 NVARCHAR(100)

)

INSERT INTO Total\_Table(col1, col2, col3, col4, col5, col6)

SELECT id, NULL, NULL, NULL, NULL, CAST(country\_name AS nvarchar(50)) FROM Country\_Table

UNION ALL

SELECT NULL, comments\_disabled, ratings\_disabled, video\_error\_or\_removed, field\_bool, NULL FROM CRVF

UNION ALL

SELECT id, NULL, NULL, NULL, NULL, day\_name FROM DOW\_Table

UNION ALL

SELECT id, NULL, NULL, NULL, NULL, time\_frame FROM TF\_Table

SELECT \* FROM Total\_Table

DROP TABLE Country\_Table

DROP TABLE CRVF

DROP TABLE DOW\_Table

DROP TABLE TF\_Table

A screenshot of a computer

Description automatically generated

## Sau khi hoàn tất tiền xử lý dữ liệu, đối với từng field trong dữ liệu, tùy thuộc kiểu dữ liệu của mỗi field, SV cần thống kê được các giá trị sau

A screenshot of a computer program

Description automatically generated

CREATE TABLE ThongKe (

FieldName NVARCHAR(200),

Mean NVARCHAR(200),

Mode NTEXT,

Median NVARCHAR(200),

Deviation NVARCHAR(200),

QTMIN NVARCHAR(200),

QT25 NVARCHAR(200),

QT50 NVARCHAR(200),

QT75 NVARCHAR(200),

QTMAX NVARCHAR(200),

)

**- Mode**

--Trending Date - Mode

WITH CTE AS (

SELECT trending\_date, COUNT(\*) AS cnt

FROM Youtube\_Test

GROUP BY trending\_date

)

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Trending Date', CAST(( SELECT STRING\_AGG(trending\_date, ' ; ') AS CombinedResult

FROM CTE

WHERE cnt = (SELECT MAX(cnt) FROM CTE)) AS NTEXT)

FROM Youtube\_Test

SELECT \* FROM ThongKe

--Channel Title - Mode

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Channel Title', (SELECT TOP 1 channel\_title

FROM Youtube\_Test

GROUP BY channel\_title

ORDER BY COUNT(\*) DESC)

--Category Id - Mode

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Category Id', CAST((SELECT TOP 1 category\_id

FROM Youtube\_Test

GROUP BY category\_id

ORDER BY COUNT(\*) DESC) AS NVARCHAR(200))

--Publish Date - Mode

WITH CTE AS (

SELECT publish\_date, COUNT(\*) AS cnt

FROM Youtube\_Test

GROUP BY publish\_date

)

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Publish Date', CAST(( SELECT STRING\_AGG(publish\_date, ' ; ') AS CombinedResult

FROM CTE

WHERE cnt = (SELECT MAX(cnt) FROM CTE)) AS NTEXT)

FROM Youtube\_Test

--Time Frame - Mode

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Time Frame', CAST((SELECT TOP 1 time\_frame

FROM Youtube\_Test

GROUP BY time\_frame

ORDER BY COUNT(\*) DESC) AS NVARCHAR(200))

--published\_day\_of\_week - Mode

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'published\_day\_of\_week', CAST((SELECT TOP 1 published\_day\_of\_week

FROM Youtube\_Test

GROUP BY published\_day\_of\_week

ORDER BY COUNT(\*) DESC) AS NVARCHAR(200))

--Publish Country - Mode

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Publish Country', CAST((SELECT TOP 1 publish\_country

FROM Youtube\_Test

GROUP BY publish\_country

ORDER BY COUNT(\*) DESC) AS NVARCHAR(200))

--Field Bool - Mode

INSERT INTO ThongKe(FieldName, Mode)

SELECT TOP 1 'Field Bool', CAST((SELECT TOP 1 field\_bool

FROM Youtube\_Test

GROUP BY field\_bool

ORDER BY COUNT(\*) DESC) AS NVARCHAR(200))

**- Mean, Mode, Median, Deviation, Min, 25, 50, 75, Max**

***Views - Mean - Mode - Median - Deviation - MIN - 25 - 50 - 75 - MAX***

WITH CTE AS (

SELECT views, COUNT(\*) AS cnt

FROM Youtube\_Test

GROUP BY views

), CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) views AS views\_q1

FROM Youtube\_Test

ORDER BY views ASC

), CTE2

AS

(

--Q2

SELECT TOP ( SELECT COUNT(\*) / 2 FROM Youtube\_Test) views AS views\_q2

FROM Youtube\_Test

ORDER BY views ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) views AS views\_q3

FROM Youtube\_Test

ORDER BY views ASC

)

INSERT INTO ThongKe(FieldName, Mean, Mode, Median, Deviation, QTMIN, QT25, QT50, QT75, QTMAX)

SELECT TOP 1 'Views', CAST((SELECT CAST(AVG(CAST(views AS BIGINT)) AS BIGINT) FROM Youtube\_Test) AS NVARCHAR(200)),

CAST((SELECT STRING\_AGG(views, ' ; ') AS CombinedResult FROM CTE WHERE cnt = (SELECT MAX(cnt) FROM CTE)) AS NTEXT),

(SELECT views FROM Youtube\_Test WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test)),

(SELECT CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(views AS FLOAT), 2)) - POWER(AVG(CAST(views AS FLOAT)), 2)) AS BIGINT) AS dlc FROM Youtube\_Test),

(SELECT ((SELECT MAX(views\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1)))),

(SELECT MAX(views\_q1) FROM CTE1), (SELECT MAX(views\_q2) FROM CTE2), (SELECT MAX(views\_q3) FROM CTE3),

(SELECT ((SELECT MAX(views\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1))))

***Likes - Mean - Mode - Median - Deviation - MIN - 25 - 50 - 75 - MAX***

WITH CTE AS (

SELECT likes, COUNT(\*) AS cnt

FROM Youtube\_Test

GROUP BY likes

), CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) likes AS views\_q1

FROM Youtube\_Test

ORDER BY likes ASC

), CTE2

AS

(

--Q2

SELECT TOP ( SELECT COUNT(\*) / 2 FROM Youtube\_Test) likes AS views\_q2

FROM Youtube\_Test

ORDER BY likes ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) likes AS views\_q3

FROM Youtube\_Test

ORDER BY likes ASC

)

INSERT INTO ThongKe(FieldName, Mean, Mode, Median, Deviation, QTMIN, QT25, QT50, QT75, QTMAX)

SELECT TOP 1 'Likes', CAST((SELECT CAST(AVG(CAST(likes AS BIGINT)) AS BIGINT) FROM Youtube\_Test) AS NVARCHAR(200)),

CAST((SELECT STRING\_AGG(likes, ' ; ') AS CombinedResult FROM CTE WHERE cnt = (SELECT MAX(cnt) FROM CTE)) AS NTEXT),

(SELECT likes FROM Youtube\_Test WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test)),

(SELECT CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(likes AS FLOAT), 2)) - POWER(AVG(CAST(likes AS FLOAT)), 2)) AS BIGINT) AS dlc FROM Youtube\_Test),

(SELECT ((SELECT MAX(views\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1)))),

(SELECT MAX(views\_q1) FROM CTE1), (SELECT MAX(views\_q2) FROM CTE2), (SELECT MAX(views\_q3) FROM CTE3),

(SELECT ((SELECT MAX(views\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1))))

***Dislikes - Mean - Mode - Median - Deviation - MIN - 25 - 50 - 75 - MAX***

WITH CTE AS (

SELECT dislikes, COUNT(\*) AS cnt

FROM Youtube\_Test

GROUP BY dislikes

), CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) dislikes AS views\_q1

FROM Youtube\_Test

ORDER BY dislikes ASC

), CTE2

AS

(

--Q2

SELECT TOP ( SELECT COUNT(\*) / 2 FROM Youtube\_Test) dislikes AS views\_q2

FROM Youtube\_Test

ORDER BY dislikes ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) dislikes AS views\_q3

FROM Youtube\_Test

ORDER BY dislikes ASC

)

INSERT INTO ThongKe(FieldName, Mean, Mode, Median, Deviation, QTMIN, QT25, QT50, QT75, QTMAX)

SELECT TOP 1 'Dislikes', CAST((SELECT CAST(AVG(CAST(dislikes AS BIGINT)) AS BIGINT) FROM Youtube\_Test) AS NVARCHAR(200)),

CAST((SELECT STRING\_AGG(dislikes, ' ; ') AS CombinedResult FROM CTE WHERE cnt = (SELECT MAX(cnt) FROM CTE)) AS NTEXT),

(SELECT dislikes FROM Youtube\_Test WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test)),

(SELECT CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(dislikes AS FLOAT), 2)) - POWER(AVG(CAST(dislikes AS FLOAT)), 2)) AS BIGINT) AS dlc FROM Youtube\_Test),

(SELECT ((SELECT MAX(views\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1)))),

(SELECT MAX(views\_q1) FROM CTE1), (SELECT MAX(views\_q2) FROM CTE2), (SELECT MAX(views\_q3) FROM CTE3),

(SELECT ((SELECT MAX(views\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1))))

***Commentscount - Mean - Mode - Median - Deviation - MIN - 25 - 50 - 75 - MAX***

WITH CTE AS (

SELECT comment\_count, COUNT(\*) AS cnt

FROM Youtube\_Test

GROUP BY comment\_count

), CTE1

AS

(

--Q1

SELECT TOP ( SELECT COUNT(\*) / 4 FROM Youtube\_Test) comment\_count AS views\_q1

FROM Youtube\_Test

ORDER BY comment\_count ASC

), CTE2

AS

(

--Q2

SELECT TOP ( SELECT COUNT(\*) / 2 FROM Youtube\_Test) comment\_count AS views\_q2

FROM Youtube\_Test

ORDER BY comment\_count ASC

), CTE3

AS

(

--Q3

SELECT TOP ( SELECT CAST(ROUND((COUNT(\*) \* 0.75), 0) AS INT) FROM Youtube\_Test) comment\_count AS views\_q3

FROM Youtube\_Test

ORDER BY comment\_count ASC

)

INSERT INTO ThongKe(FieldName, Mean, Mode, Median, Deviation, QTMIN, QT25, QT50, QT75, QTMAX)

SELECT TOP 1 'Comment Count', CAST((SELECT CAST(AVG(CAST(comment\_count AS BIGINT)) AS BIGINT) FROM Youtube\_Test) AS NVARCHAR(200)),

CAST((SELECT STRING\_AGG(comment\_count, ' ; ') AS CombinedResult FROM CTE WHERE cnt = (SELECT MAX(cnt) FROM CTE)) AS NTEXT),

(SELECT comment\_count FROM Youtube\_Test WHERE [index] IN (SELECT COUNT(\*) / 2 FROM Youtube\_Test)),

(SELECT CAST(SQRT((1.0 / COUNT(\*)) \* SUM(POWER(CAST(comment\_count AS FLOAT), 2)) - POWER(AVG(CAST(comment\_count AS FLOAT)), 2)) AS BIGINT) AS dlc FROM Youtube\_Test),

(SELECT ((SELECT MAX(views\_q1) FROM CTE1) - 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1)))),

(SELECT MAX(views\_q1) FROM CTE1), (SELECT MAX(views\_q2) FROM CTE2), (SELECT MAX(views\_q3) FROM CTE3),

(SELECT ((SELECT MAX(views\_q3) FROM CTE3) + 1.5 \* ((SELECT MAX(views\_q3) FROM CTE3) - (SELECT MAX(views\_q1) FROM CTE1))))

**QUÁ TRÌNH TIỀN XỬ LÝ DỮ LIỆU - PYTHON**

* 1. **Làm sạch dữ liệu**

**Clone dataset**

import kagglehub

# Download latest version

path = kagglehub.dataset\_download("thedevastator/youtube-trending-videos-dataset")

print("Path to dataset files:", path)

**Khởi tạo**

import pandas as pd

import numpy as np

import csv

import random

import matplotlib.pyplot as plt

import seaborn as sns

from joypy import joyplot

from matplotlib.cm import viridis

import plotly.express as px

YT\_Table = pd.read\_csv('E:\Data Analysis Basic Method\Project\Data-Analysis-Basic-Method\Dataset5\Youtube\_Test.csv')

YT\_Table

DF = pd.DataFrame(YT\_Table)

DF

**A screenshot of a computer

Description automatically generated**

* + 1. **Điền giá trị còn thiếu**
* Chủ động xóa các fields

Nếu dữ liệu không có giá trị NULL, SV chủ động xóa trên tất cả các field (ngoại trừ field ID), mỗi field khoảng 5 giá trị (trên 5 record liền nhau để dễ quan sát kết quả).

A screen shot of a computer

Description automatically generated

***Trending\_date***

DF = DF.sort\_values(by='index', ascending=True)

DF.loc[[i for i in range(0,5)], 'trending\_date'] = '17.14.111'

DF.loc[[i for i in range(5,10)], 'trending\_date'] = '17.144.11'

DF.loc[[i for i in range(10,15)], 'trending\_date'] = '177.14.11'

DF.loc[[i for i in range(15,20)], 'trending\_date'] = '11.14.17'

DF.loc[[i for i in range(20,25)], 'trending\_date'] = '14.11.17'

DF

***Views***

DF.loc[[i for i in range(5,9)], 'views'] = np.nan

***Likes***

DF.loc[[i for i in range(9, 15)], 'likes'] = np.nan

***Dislikes***

DF.loc[[i for i in range(15, 20)], 'dislikes'] = np.nan

***Comment\_count***

DF.loc[[i for i in range(20, 25)], 'comment\_count'] = np.nan

***Comments\_disabled***

DF.loc[[i for i in range(25, 30)], 'comments\_disabled'] = np.nan

***Ratings\_disabled***

DF.loc[[i for i in range(30, 35)], 'ratings\_disabled'] = np.nan

***Video\_error\_or\_removed***

DF.loc[[i for i in range(35, 40)], 'video\_error\_or\_removed'] = np.nan

***Tags***

DF.loc[[i for i in range(0,5)], 'tags'] = np.nan

* Điền các giá trị vào fields NULL

Không điền những giá trị đơn giản (như “không biết”, “không có”, hay min, max, mean, mode, median) cho những giá trị NULL mà cần điền bằng những giá trị đòi hỏi tính toán phức tạp hơn, ví dụ điền giá trị NULL bằng giá trị của mode, nếu có nhiều mode thì chọn mode có giá trị thấp nhất hoặc đối với kiểu số thực, có thể thực hiện làm tròn (đến phần nguyên hoặc phần ngàn, …) trước khi tính mean/mode/median.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

***Views***

MMM = [int(DF['views'].mean()), int(DF['views'].mode()[0]), int(DF['views'].median())]

DF['views'].fillna(min(MMM), inplace=True)

***Likes***

MMM = [int(DF['likes'].mean()), int(DF['likes'].mode()[0]), int(DF['likes'].median())]

DF['likes'].fillna(min(MMM), inplace=True)

***Dislikes***

MMM = [int(DF['dislikes'].mean()), int(DF['dislikes'].mode()[0]), int(DF['dislikes'].median())]

DF['dislikes'].fillna(min(MMM), inplace=True)

***Comment\_count***

MMM = [int(DF['comment\_count'].mean()), int(DF['comment\_count'].mode()[0]), int(DF['comment\_count'].median())]

DF['comment\_count'].fillna(min(MMM), inplace=True)

***Comments\_disabled***

true\_ratio = len(DF[DF['comments\_disabled'] == True]) / len(DF)

random\_fill = np.random.choice([True, False], size=len(DF), p=[true\_ratio, 1 - true\_ratio])

DF['comments\_disabled'] = DF['comments\_disabled'].fillna(pd.Series(random\_fill))

***Ratings\_disabled***

true\_ratio = len(DF[DF['ratings\_disabled'] == True]) / len(DF)

random\_fill = np.random.choice([True, False], size=len(DF), p=[true\_ratio, 1 - true\_ratio])

DF['ratings\_disabled'] = DF['ratings\_disabled'].fillna(pd.Series(random\_fill))

***Video\_error\_or\_removed***

true\_ratio = len(DF[DF['video\_error\_or\_removed'] == True]) / len(DF)

random\_fill = np.random.choice([True, False], size=len(DF), p=[true\_ratio, 1 - true\_ratio])

DF['video\_error\_or\_removed'] = DF['video\_error\_or\_removed'].fillna(pd.Series(random\_fill))

***Tags***

def create\_tags(row):

    tags = ''

    lst = row['title'].split()

    for j, word in enumerate(lst):

        if j == 0:

            tags += str(word) + '|"'

        elif j == len(lst) - 1:

            tags += str(word) + '"""'

        else:

            tags += str(word) + '""|""'

    tags = tags.strip()

    tags = tags.translate(str.maketrans('', '', ":()?!"))

    return tags

DF['tags'] = DF.apply(lambda row: create\_tags(row) if pd.isna(row['tags']) else row['tags'], axis=1)

***Tags[None]***

DF\_2 = DF[DF['tags'] == '[none]']

DF\_2.reset\_index(drop=True, inplace=True)

for i in range(len(DF\_2)):

    lst = DF\_2.loc[i, 'title'].split()

    tags\_list = []

    for j in range(len(lst)):

        if j == 0:

            tags\_list.append(str(lst[j]) + '|"')

        elif j == len(lst) - 1:

            tags\_list.append(str(lst[j]) + '"""')

        else:

            tags\_list.append(str(lst[j]) + '""|""')

    tags = ''.join(tags\_list).strip()

    for char in ":()?!":

        tags = tags.replace(char, "")

    DF\_2.loc[i, 'tags'] = tags

DF.loc[DF['tags'] == '[none]', ['title', 'tags']] = DF\_2[['title', 'tags']].values

* + 1. **Làm mịn dữ liệu**

Chọn thuộc tính có miền giá trị cao nhất trong dữ liệu. Thực hiện chia miền giá trị này

thành nhiều nhóm (>=5 nhóm). Chọn 1 trong 3 cách sau để làm mịn dữ liệu:

 Thay thế bằng giá trị trung bình (mean) của nhóm.

 Thay thế bằng giá trị trung vị (median) của nhóm.

 Làm mịn theo ranh giới của nhóm (smoothing by bin boundaries)

* Chia thành 5 nhóm và theo 3 cách khác nhau
* Mean

A screen shot of a computer

Description automatically generated

DF = DF.sort\_values(by='views', ascending=True)

DF1 = DF[:int(len(DF)/5)]

DF2 = DF[int(len(DF)/5):int(len(DF)/5)\*2]

DF3 = DF[int(len(DF)/5)\*2:int(len(DF)/5)\*3]

DF4 = DF[int(len(DF)/5)\*3:int(len(DF)/5)\*4]

DF5 = DF[int(len(DF)/5)\*4:]

DF.loc[[i for i in range(0,int(len(DF)/5))], 'views'] = int(DF1['views'].mean())

DF.loc[[i for i in range(int(len(DF)/5),int(len(DF)/5)\*2)], 'views'] = int(DF2['views'].mean())

DF.loc[[i for i in range(int(len(DF)/5)\*2,int(len(DF)/5)\*3)], 'views'] = int(DF3['views'].mean())

DF.loc[[i for i in range(int(len(DF)/5)\*3,int(len(DF)/5)\*4)], 'views'] = int(DF4['views'].mean())

DF.loc[[i for i in range(int(len(DF)/5)\*4,len(DF))], 'views'] = int(DF5['views'].mean())

DF

* Median

A screenshot of a computer

Description automatically generated

DF = DF.sort\_values(by='likes', ascending=True)

DF1 = DF[:int(len(DF)/5)]

DF2 = DF[int(len(DF)/5):int(len(DF)/5)\*2]

DF3 = DF[int(len(DF)/5)\*2:int(len(DF)/5)\*3]

DF4 = DF[int(len(DF)/5)\*3:int(len(DF)/5)\*4]

DF5 = DF[int(len(DF)/5)\*4:]

DF.loc[[i for i in range(0,int(len(DF)/5))], 'likes'] = int(DF1['likes'].median())

DF.loc[[i for i in range(int(len(DF)/5),int(len(DF)/5)\*2)], 'likes'] = int(DF2['likes'].median())

DF.loc[[i for i in range(int(len(DF)/5)\*2,int(len(DF)/5)\*3)], 'likes'] = int(DF3['likes'].median())

DF.loc[[i for i in range(int(len(DF)/5)\*3,int(len(DF)/5)\*4)], 'likes'] = int(DF4['likes'].median())

DF.loc[[i for i in range(int(len(DF)/5)\*4,len(DF))], 'likes'] = int(DF5['likes'].median())

DF

* Boundaries

DF = DF.sort\_values(by='dislikes', ascending=True)

DF1 = DF[:int(len(DF)/5)]

DF2 = DF[int(len(DF)/5):int(len(DF)/5)\*2]

DF3 = DF[int(len(DF)/5)\*2:int(len(DF)/5)\*3]

DF4 = DF[int(len(DF)/5)\*3:int(len(DF)/5)\*4]

DF5 = DF[int(len(DF)/5)\*4:]

for i in range(0,int(len(DF)/5)):

    if i - int(DF1['dislikes'].min()) < int(DF1['dislikes'].max()) - i:

        DF.loc[i, 'dislikes'] = int(DF1['dislikes'].min())

    else:

        DF.loc[i, 'dislikes'] = int(DF1['dislikes'].max())

for i in range(int(len(DF)/5),int(len(DF)/5)\*2):

    if i - int(DF2['dislikes'].min()) < int(DF2['dislikes'].max()) - i:

        DF.loc[i, 'dislikes'] = int(DF2['dislikes'].min())

    else:

        DF.loc[i, 'dislikes'] = int(DF2['dislikes'].max())

for i in range(int(len(DF)/5)\*2,int(len(DF)/5)\*3):

    if i - int(DF3['dislikes'].min()) < int(DF3['dislikes'].max()) - i:

        DF.loc[i, 'dislikes'] = int(DF3['dislikes'].min())

    else:

        DF.loc[i, 'dislikes'] = int(DF3['dislikes'].max())

for i in range(int(len(DF)/5)\*3,int(len(DF)/5)\*4):

    if i - int(DF4['dislikes'].min()) < int(DF4['dislikes'].max()) - i:

        DF.loc[i, 'dislikes'] = int(DF4['dislikes'].min())

    else:

        DF.loc[i, 'dislikes'] = int(DF4['dislikes'].max())

for i in range(int(len(DF)/5)\*4,len(DF)):

    if i - int(DF5['dislikes'].min()) < int(DF5['dislikes'].max()) - i:

        DF.loc[i, 'dislikes'] = int(DF5['dislikes'].min())

    else:

        DF.loc[i, 'dislikes'] = int(DF5['dislikes'].max())

DF

A screenshot of a computer

Description automatically generated

Chọn field có độ lệch chuẩn lớn nhất trong dữ liệu: thực hiện tăng 10% giá trị của thuộc tính này cho 5 giá trị nhỏ nhất và giảm 10% giá trị của thuộc tính này cho 5 giá trị lớn nhất.

S2 = (DF['likes']\*\*2).sum()

S3 = (DF['dislikes']\*\*2).sum()

S4 = (DF['comment\_count']\*\*2).sum()

y = S2 / len(DF) - DF['likes'].mean()\*\*2

z = S3 / len(DF) - DF['dislikes'].mean()\*\*2

t = S4 / len(DF) - DF['comment\_count'].mean()\*\*2

variance = sum((DF['views'] - DF['views'].mean())\*\*2) / len(DF)

std\_views = variance\*\*0.5

std\_likes = y\*\*0.5

std\_dislikes = z\*\*0.5

std\_comment\_count = t\*\*0.5

print(std\_views, std\_likes, std\_dislikes, std\_comment\_count)

DFS\_SORT = DF.sort\_values(by='views', ascending=True)

DFS\_SORT.iloc[:5, DFS\_SORT.columns.get\_loc('views')] \*= 1.1

DFS\_SORT.iloc[-5:, DFS\_SORT.columns.get\_loc('views')] \*= 0.9

DF = DFS\_SORT

* + 1. **Xác định giá trị ngoại lệ**

Xác định dữ liệu có chứa giá trị ngoại lệ (tính theo five number summary) của từng thuộc tính số hay không?

***--Views***

***A screenshot of a computer

Description automatically generated***

DFSV = DF.sort\_values(by='views', ascending=True)

Q1 = DFSV.iloc[int(len(DF)\*0.25)]['views']

Q3 = DFSV.iloc[int(len(DF)\*0.75)]['views']

OL\_Views = DFSV[(DFSV['views'] < Q1 - (1.5 \* (Q3 - Q1))) | (DFSV['views'] > Q3 + (1.5 \* (Q3 - Q1)))]

OL\_Views

***--Likes***

A screenshot of a computer

Description automatically generated

DFSL = DF.sort\_values(by='likes', ascending=True)

Q1 = DFSL.iloc[int(len(DF)\*0.25)]['likes']

Q3 = DFSL.iloc[int(len(DF)\*0.75)]['likes']

OL\_Likes = DFSL[(DFSL['likes'] < Q1 - (1.5 \* (Q3 - Q1))) | (DFSL['likes'] > Q3 + (1.5 \* (Q3 - Q1)))]

OL\_Likes

***--Dislikes***

***A screenshot of a computer

Description automatically generated***

DFSDL = DF.sort\_values(by='dislikes', ascending=True)

Q1 = DFSDL.iloc[int(len(DF)\*0.25)]['dislikes']

Q3 = DFSDL.iloc[int(len(DF)\*0.75)]['dislikes']

OL\_DL = DFSDL[(DFSDL['dislikes'] < Q1 - (1.5 \* (Q3 - Q1))) | (DFSDL['dislikes'] > Q3 + (1.5 \* (Q3 - Q1)))]

OL\_DL

***--Comment Count***

***A screenshot of a computer

Description automatically generated***

DFSCC = DF.sort\_values(by='comment\_count', ascending=True)

Q1 = DFSCC.iloc[int(len(DF)\*0.25)]['comment\_count']

Q3 = DFSCC.iloc[int(len(DF)\*0.75)]['comment\_count']

OL\_CC = DFSCC[(DFSCC['comment\_count'] < Q1 - (1.5 \* (Q3 - Q1))) | (DFSCC['comment\_count'] > Q3 + (1.5 \* (Q3 - Q1)))]

OL\_CC

* + 1. **Xử lý dữ liệu không nhất quán**
* Đối với field có kiểu dữ liệu ngày: từ file dữ liệu gốc, SV thực hiện đổi kiểu dữ liệu thành chuỗi, sau đó sửa thủ công các giá trị ngày cho ít nhất 5 giá trị để có 5 dữ liệu có định dạng ngày khác với các record còn lại. Ví dụ định dạng của ngày hiện tại là mm/dd/yy, sửa 5 giá trị để 5 giá trị này có kiểu cùng là dd/mm/yy hoặc cùng có kiểu là yy/mm/dd. Sau đó thực hiện cập nhật lại giá trị cho 5 giá trị này.

***Trending\_date***

***A screenshot of a computer

Description automatically generated***

DF.loc[(DF['trending\_date'] == '17.14.111') | (DF['trending\_date'] == '17.144.11') | (DF['trending\_date'] == '177.14.11') | (DF['trending\_date'] == '11.14.17') | (DF['trending\_date'] == '14.11.17'), 'trending\_date'] = '17.14.11'

***Đồng bộ -> dd/mm/yyyy***

***A screenshot of a computer screen

Description automatically generated***

def convert\_date(date\_str):

    year, day, month = date\_str.split('.')

    year = '20' + year

    return f'{day}/{month}/{year}'

DF['trending\_date'] = DF['trending\_date'].apply(convert\_date)

* Đối với field kiểu danh nghĩa, SV cũng chủ động chỉnh sửa 5 giá trị, sau đó dủng lệnh thực hiện hiệu chỉnh lại cho đúng với giá trị gốc

A screenshot of a computer

Description automatically generated

--published\_day\_of\_week, publish\_country

DF.loc[[i for i in range(8,13)], 'published\_day\_of\_week'] = 'MONDAYY'

DF.loc[[i for i in range(0,5)], 'publish\_country'] = 'uss'

--MONDAYY -> Monday, uss -> US

valid\_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

DF.loc[~DF['published\_day\_of\_week'].isin(valid\_days), 'published\_day\_of\_week'] = DF['published\_day\_of\_week'].str.title()

import re

def correct\_day(day):

    if day in valid\_days:

        return day

    corrected\_day = re.sub(r'(\w)\1+', r'\1', day)

    return corrected\_day if corrected\_day in valid\_days else day.lower()

DF['published\_day\_of\_week'] = DF['published\_day\_of\_week'].apply(correct\_day)

DF

--Publish\_country

valid\_countrys = ['US', 'FRANCE', 'CANADA', 'GB']

DF.loc[~DF['publish\_country'].isin(valid\_days), 'publish\_country'] = DF['publish\_country'].str.upper()

def correct\_country(country):

    if country in valid\_countrys:

        return country

    corrected\_country = re.sub(r'(\w)\1+', r'\1', country)

    return corrected\_country if corrected\_country in valid\_countrys else country.lower()

DF['publish\_country'] = DF['publish\_country'].apply(correct\_country)

A screenshot of a computer

Description automatically generated

* 1. **Tích hợp dữ liệu**
  2. **Giảm thiểu dữ liệu**
* PCA

-- PCA - Giam so chieu tu 4(views, likes, dislikes, comment\_count) -> 3 | 2 | 1

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(DF[['views', 'likes', 'dislikes', 'comment\_count']])

pca = PCA(n\_components=2)

data\_pca = pca.fit\_transform(data\_scaled)

print("Dữ liệu sau khi PCA:", data\_pca)

print("Tỷ lệ phương sai giải thích:", pca.explained\_variance\_ratio\_)

data\_pca

* Phân cụm

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

plt.scatter(DF['views'], DF['likes'], c='gray', edgecolor='k')

plt.title("Dữ liệu trước khi phân cụm")

plt.xlabel("X")

plt.ylabel("Y")

plt.show()

A graph of a graph showing a pointy object

Description automatically generated with medium confidence

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=3, random\_state=42)

kmeans.fit(DF[['views', 'likes']])

DF['Cluster'] = kmeans.labels\_

print("\nDữ liệu sau phân cụm:")

print(DF.head())

plt.scatter(DF['views'], DF['likes'], c=DF['Cluster'], cmap='viridis', edgecolor='k')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=300, c='red')

plt.title("Kết quả phân cụm với K-Means")

plt.xlabel("X")

plt.ylabel("Y")

plt.show()

A diagram of a bird

Description automatically generated with medium confidence

* 1. **Chuyển đổi và phân tách dữ liệu**

- Kiểu danh mục (ví dụ như các mùa: Xuân, Hạ, Thu, Đông): chuyển thành kiểu số nguyên.

- Kiểu thứ tự (ví dụ như: Giỏi, khá, trung bình, yếu): chuyển thành kiểu số nguyên. - Kiểu boolean: khi dữ liệu có từ 5 field kiểu này trở lên, chuyển 5 field thành 1 field kiểu số.

- Kiểu chuỗi: trong field kiểu chuỗi lại chứa danh sách các chuỗi con có ý nghĩa (ví dụ Tên hàng Đã mua trong đó gồm tên các sản phẩm đã mua trong đơn hàng như: bột giặt, khăn tắm, chén, khăn tắm): cần tách tên các sản phẩm ra khỏi field này để có nhiều field mới. Sau đó, do có thể có rất nhiều field mới nên cần chuyển đổi tất cả các field mới này trở lại thành 1 field duy nhất có kiểu là int.

Trong quá trình chuyển đổi có yêu cầu tạo ra các table làm trung gian, SV cần tìm cách gộp các table vừa phát sinh có cùng cấu trúc, ý nghĩa để giảm bớt số lượng table thực tế sẽ dùng.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

***publish\_country***

country = DF['publish\_country'].drop\_duplicates()

dict\_country = {country: i for i, country in enumerate(country, start=1)}

DF['publish\_country'] = DF['publish\_country'].map(dict\_country)

***published\_day\_of\_week***

ordered\_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

DF['published\_day\_of\_week'] = pd.Categorical(DF['published\_day\_of\_week'], categories=ordered\_days, ordered=True)

sorted\_days = DF['published\_day\_of\_week'].drop\_duplicates().sort\_values()

dict\_day = {day: i + 1 for i, day in enumerate(sorted\_days, start=1)}

print(dict\_day)

DF['published\_day\_of\_week'] = DF['published\_day\_of\_week'].map(dict\_day)

***time\_frame***

times = [f'{i}:00 to {i}:59' for i in range(0, 24)]

DF['time\_frame'] = pd.Categorical(DF['time\_frame'], categories=times, ordered=True)

time\_frame = DF['time\_frame'].drop\_duplicates().sort\_values()

dict\_time\_frame = {time: i for i, time in enumerate(time\_frame, start=1)}

DF['time\_frame'] = DF['time\_frame'].map(dict\_time\_frame)

***Kieu boolean***

dict\_bool = {True: 1, False: 0}

DF['comments\_disabled'] = DF['comments\_disabled'].map(dict\_bool)

DF['ratings\_disabled'] = DF['ratings\_disabled'].map(dict\_bool)

DF['video\_error\_or\_removed'] = DF['video\_error\_or\_removed'].map(dict\_bool)

***Gộp thành 1 field***

DF.insert(column= 'bool\_field', loc= len(DF.columns), value=np.nan)

***Gán giá trị***

unique\_combinations = DF[['comments\_disabled', 'ratings\_disabled', 'video\_error\_or\_removed']].drop\_duplicates()

DF['bool\_field'] = DF.apply(lambda row: f"{row['comments\_disabled']}{row['ratings\_disabled']}{row['video\_error\_or\_removed']}" if pd.isna(row['bool\_field']) else row['bool\_field'], axis=1)

***Xóa các cột***

DFT = DFT.pop('comments\_disabled')

DFT = DFT.pop('ratings\_disabled')

DFT = DFT.pop('video\_error\_or\_removed')

***Trong quá trình chuyển đổi có yêu cầu tạo ra các table làm trung gian, SV cần tìm cách gộp các table vừa phát sinh có cùng cấu trúc, ý nghĩa để giảm bớt số lượng table thực tế sẽ dùng.***

* 1. **Sau khi hoàn tất tiền xử lý dữ liệu, đối với từng field trong dữ liệu, tùy thuộc kiểu dữ liệu của mỗi field, SV cần thống kê được các giá trị sau**

**- Mode**

**A screen shot of a black screen

Description automatically generated**

-- trending\_date, channel\_title, category\_id, publish\_date, time\_frame, published\_day\_of\_week, publish\_country, views, likes, dislikes, comment\_count

DF[['trending\_date', 'channel\_title', 'category\_id', 'publish\_date', 'time\_frame', 'published\_day\_of\_week', 'publish\_country', 'views', 'likes', 'dislikes', 'comment\_count']].mode()

**- Mean, Mode, Median, Deviation, Min, 25, 50, 75, Max**

***views, likes, dislikes, comment\_count -> describle***

DF[['views', 'likes', 'dislikes', 'comment\_count']].describe()

A screenshot of a black screen

Description automatically generated

Trực quan hóa dữ liệu – Python, R

****Python library****

1. Pandas (import pandas as pd)

* **Mục đích:** Pandas là một thư viện mạnh mẽ dùng để xử lý và phân tích dữ liệu. Pandas giúp bạn dễ dàng thao tác với dữ liệu dạng bảng (DataFrame) và thực hiện các thao tác như lọc, nhóm, tóm tắt, và tính toán.
* **Ứng dụng trong trực quan hóa:**
  + Chuẩn bị và làm sạch dữ liệu (ví dụ: xóa dữ liệu thiếu, thay đổi kiểu dữ liệu).
  + Tạo DataFrame từ các nguồn dữ liệu khác nhau (CSV, Excel, SQL, JSON).

**2. Numpy (import numpy as np)**

* **Mục đích:** Numpy là thư viện toán học mạnh mẽ, chủ yếu cung cấp các cấu trúc dữ liệu đa chiều và các phép toán toán học. Nó hỗ trợ nhanh chóng các tính toán khoa học và kỹ thuật.
* **Ứng dụng trong trực quan hóa:**
  + Tạo dữ liệu giả lập (ngẫu nhiên).
  + Thực hiện các phép toán mảng để xử lý dữ liệu trước khi vẽ đồ thị.

**3. Matplotlib (import matplotlib.pyplot as plt)**

* **Mục đích:** Matplotlib là thư viện chính để tạo các đồ thị cơ bản trong Python. Nó cung cấp các công cụ để vẽ biểu đồ 2D như line charts, bar charts, scatter plots, pie charts, và nhiều loại biểu đồ khác.
* **Ứng dụng trong trực quan hóa:**
  + Vẽ các biểu đồ cơ bản như line chart, scatter plot, bar chart, và histogram.
  + Tùy chỉnh các tham số trực quan như màu sắc, tiêu đề, nhãn trục.

**4. Seaborn (import seaborn as sns)**

* **Mục đích:** Seaborn được xây dựng dựa trên Matplotlib và cung cấp các biểu đồ đẹp mắt, dễ sử dụng hơn, đặc biệt là cho các loại dữ liệu thống kê như heatmaps, violin plots, và boxplots.
* **Ứng dụng trong trực quan hóa:**
  + Tạo các biểu đồ thống kê cao cấp như heatmap, pairplot, violin plot.
  + Làm việc tốt với dữ liệu Pandas và hỗ trợ các chủ đề màu sắc đẹp mắt.

**5. Joypy (import joypy)**

* **Mục đích:** Joypy là một thư viện cho phép tạo các **Joy plots** (hay còn gọi là **Ridgeline plots**), giúp hình dung phân bố của một biến trên các nhóm khác nhau.
* **Ứng dụng trong trực quan hóa:**
  + Trực quan hóa phân bố dữ liệu của nhiều nhóm trong cùng một biểu đồ.
  + Thường dùng để vẽ các biểu đồ phân bố nhiều chiều cho dữ liệu liên quan đến phân phối.

**6. Matplotlib.cm (import matplotlib.cm as viridis)**

* **Mục đích:** Thư viện cm của Matplotlib chứa các bảng màu. viridis là một trong các bảng màu phổ biến và được chọn vì sự dễ đọc và rõ ràng của nó.
* **Ứng dụng trong trực quan hóa:**
  + Sử dụng bảng màu để trực quan hóa các giá trị liên tục trong các biểu đồ.

**7. Plotly (import plotly.express as px)**

* **Mục đích:** Plotly là một thư viện mạnh mẽ để tạo các biểu đồ tương tác, đặc biệt là cho các biểu đồ 3D và trên bản đồ.
* **Ứng dụng trong trực quan hóa:**
  + Tạo các biểu đồ tương tác đẹp mắt với khả năng zoom, hover và click.
  + Phù hợp cho các ứng dụng web hoặc trực quan hóa dữ liệu trên nền tảng trực tuyến.

R library

- R là một ngôn ngữ lập trình và môi trường phần mềm dành cho phân tích thống kê, tính toán số học, và đồ họa. Đây là một công cụ mạnh mẽ cho các nhà phân tích dữ liệu, nhà khoa học dữ liệu, và các nhà nghiên cứu, giúp xử lý và trực quan hóa dữ liệu hiệu quả. Dưới đây là một số thư viện phổ biến trong R mà bạn đã cài đặt và sử dụng:

* **Tidyverse** là một bộ các gói phần mềm trong R được thiết kế để giúp người dùng thao tác với dữ liệu dễ dàng hơn. Tidyverse bao gồm các gói cơ bản như ggplot2, dplyr, tidyr, readr, và lubridate. Những gói này giúp người dùng dễ dàng xử lý, phân tích, và trực quan hóa dữ liệu.
* **ggplot2** là một thư viện đồ họa mạnh mẽ trong R, được phát triển bởi Hadley Wickham. Nó sử dụng **Grammar of Graphics** để xây dựng các biểu đồ một cách linh hoạt và tùy chỉnh. ggplot2 rất phổ biến trong việc tạo ra các biểu đồ đẹp mắt và dễ hiểu.
* **dplyr** là một phần của tidyverse, cung cấp các chức năng mạnh mẽ để thao tác với dữ liệu như lọc, nhóm, sắp xếp, và tính toán. Các hàm chủ yếu như filter(), mutate(), select(), và arrange() giúp xử lý dữ liệu hiệu quả.
* **lubridate** là một thư viện trong R dùng để xử lý và phân tích các dữ liệu liên quan đến thời gian. Các hàm trong lubridate giúp làm việc với các định dạng ngày tháng dễ dàng hơn, ví dụ như ymd(), mdy(), hour(), minute().
* **hrbrthemes** cung cấp các chủ đề (themes) đẹp mắt và dễ tùy chỉnh cho các biểu đồ trong ggplot2. Thư viện này giúp cải thiện hình thức của các biểu đồ bằng cách sử dụng các màu sắc và phông chữ phù hợp.
* **tidyr** là một thư viện giúp chuyển đổi dữ liệu từ dạng rộng sang dạng dài và ngược lại, giúp dữ liệu dễ dàng hơn trong việc phân tích. Các hàm như gather(), spread(), và separate() rất hữu ích trong việc xử lý dữ liệu.
* **viridis** cung cấp một bảng màu (color scale) đẹp và dễ nhìn cho các biểu đồ, đặc biệt là khi làm việc với dữ liệu có giá trị liên tục. Nó rất hữu ích trong việc tạo ra các biểu đồ dễ tiếp cận và có thể đọc được, kể cả cho người khiếm thị màu sắc.
* **plotly** là một thư viện mạnh mẽ để tạo các biểu đồ tương tác trong R. Nó cho phép người dùng tạo các biểu đồ động (interactive charts) và chia sẻ chúng trực tuyến. plotly hỗ trợ nhiều loại biểu đồ như biểu đồ scatter, bar, line, và heatmap.
* **gapminder** là một thư viện cung cấp bộ dữ liệu nổi tiếng về các yếu tố như GDP, tuổi thọ, và dân số của các quốc gia trong khoảng thời gian dài. Bộ dữ liệu này được sử dụng rộng rãi trong các ví dụ về phân tích và trực quan hóa dữ liệu.
* **ggridges** cung cấp các hàm để tạo biểu đồ **ridgeline**, biểu đồ phân bố của nhiều nhóm, cho phép dễ dàng so sánh sự phân bố của các biến trong các nhóm khác nhau.
* **GGally** là một thư viện mở rộng ggplot2 cung cấp các hàm bổ sung cho việc tạo các biểu đồ phân tích mối quan hệ giữa các biến. Một trong những công cụ phổ biến là ggpairs(), giúp tạo ra một ma trận các biểu đồ phân tán và các biểu đồ xác suất cho tất cả các cặp biến trong một bộ dữ liệu.

## Scatter chart

 Biểu đồ scatter (hay biểu đồ phân tán) là một loại biểu đồ thể hiện mối quan hệ giữa hai biến số bằng cách sử dụng các điểm trong không gian tọa độ.

 Trục **X** biểu diễn giá trị của một biến số (biến độc lập), và trục **Y** biểu diễn giá trị của biến số khác (biến phụ thuộc). Mỗi điểm trên biểu đồ tương ứng với một cặp giá trị (x,y)(x, y)(x,y).

**Biểu đồ scatter giữa số lượng "likes" và "views" trên YouTube:**

* Trục X: Số lượng "likes".
* Trục Y: Số lượng "views".
* Ý nghĩa: Biểu đồ giúp phát hiện liệu số lượng "likes" có tỷ lệ thuận với "views" hay không.

A graph with orange dots

Description automatically generated

## Barplot chart

Biểu đồ **Barplot** (hay còn gọi là **Bar Chart**) là một loại biểu đồ được sử dụng phổ biến trong trực quan hóa dữ liệu để so sánh các giá trị của các nhóm khác nhau. Các cột trong biểu đồ có thể có chiều cao (hoặc chiều dài, nếu biểu đồ nằm ngang) tương ứng với giá trị của mỗi nhóm. Đây là một cách hiệu quả để so sánh các giá trị định lượng, đặc biệt khi các nhóm là rời rạc hoặc không liên tục.

**Cấu trúc và Ý nghĩa của Biểu đồ Barplot**

* **Trục x:** Đại diện cho các nhóm hoặc danh mục cần so sánh (trong trường hợp này là các quốc gia xuất bản video).
* **Trục y:** Đại diện cho giá trị của mỗi nhóm (ở đây là số lượt xem, views).
* **Các thanh (bars):** Chiều cao (hoặc chiều dài đối với bar chart ngang) của các thanh biểu diễn giá trị của mỗi nhóm. Thanh càng cao, giá trị càng lớn.

A bar chart with blue bars

Description automatically generated

Biểu đồ này giúp bạn so sánh tổng số lượt xem giữa các quốc gia xuất bản video. Mỗi cột trong biểu đồ đại diện cho một quốc gia, và chiều cao của cột thể hiện tổng số lượt xem mà quốc gia đó có. Điều này giúp bạn dễ dàng nhận thấy quốc gia nào có lượt xem cao nhất và quốc gia nào có lượt xem thấp nhất, giúp rút ra các nhận định về sự phổ biến của video trên các quốc gia khác nhau.

## Pie chart

Biểu đồ **Pie Chart** (Biểu đồ hình tròn) là một dạng biểu đồ được sử dụng để trực quan hóa tỷ lệ phần trăm của các phần trong một tổng thể. Mỗi phần trong biểu đồ được biểu diễn dưới dạng một lát cắt của hình tròn, và kích thước của mỗi lát cắt tỷ lệ với tỷ lệ phần trăm của phần đó trong tổng thể. Biểu đồ này rất hiệu quả khi bạn muốn minh họa tỷ lệ hoặc phân bố các phần trong một tổng thể.

A pie chart with numbers and a few days of the week

Description automatically generated

Biểu đồ Pie Chart này giúp bạn phân tích và hiểu tỷ lệ phần trăm của số lượt xem video (views) theo từng ngày trong tuần mà video được xuất bản. Mỗi lát cắt trong hình tròn đại diện cho một ngày trong tuần, và kích thước của lát cắt tỷ lệ với tổng số lượt xem cho ngày đó. Việc sử dụng Pie Chart giúp bạn dễ dàng nhận biết ngày nào có lượt xem cao nhất và ngày nào có lượt xem thấp nhất.

## Doughnut chart

Biểu đồ **Doughnut Chart** là một dạng đặc biệt của biểu đồ hình tròn (Pie Chart), trong đó phần trung tâm của biểu đồ được cắt bỏ, tạo thành một hình tròn rỗng giống như một chiếc bánh doughnut. Biểu đồ này giúp trực quan hóa tỷ lệ của các phần trong tổng thể, tương tự như Pie Chart, nhưng với không gian ở trung tâm để có thể thêm các thông tin bổ sung hoặc tạo một cảm giác rõ ràng hơn.

A pie chart with different colored circles

Description automatically generated

Biểu đồ Doughnut Chart này giúp bạn trực quan hóa tỷ lệ phần trăm số lượt xem video theo các quốc gia xuất bản. Mỗi lát cắt trong biểu đồ đại diện cho một quốc gia, với kích thước của lát cắt tỷ lệ với số lượt xem của quốc gia đó. Việc sử dụng Doughnut Chart không chỉ giúp bạn quan sát tỷ lệ giữa các quốc gia mà còn tạo ra một hình thức trực quan dễ nhìn và thẩm mỹ hơn so với Pie Chart truyền thống.

## Line chart

Biểu đồ **Line Chart** là một loại biểu đồ được sử dụng để thể hiện sự thay đổi của một biến theo thời gian hoặc theo một chuỗi số liệu liên tục. Trong trường hợp này, Line Chart được sử dụng để thể hiện **lượt xem trung bình** theo **thời gian xuất bản (time frame)**.

A line chart with blue line

Description automatically generated

Biểu đồ **Line Chart** này giúp bạn quan sát sự thay đổi của lượt xem trung bình theo từng thời gian xuất bản (time\_frame). Mỗi điểm trên đường thẳng thể hiện lượt xem trung bình tại một thời gian xuất bản cụ thể. Đường thẳng nối các điểm cho thấy xu hướng thay đổi của lượt xem theo thời gian.

## Violin chart

Biểu đồ **Violin Chart** kết hợp các yếu tố của biểu đồ hộp (Boxplot) và mật độ phân phối (Density Plot) để cung cấp cái nhìn sâu sắc hơn về phân phối của dữ liệu. Trong trường hợp này, biểu đồ violin được sử dụng để trực quan hóa sự phân bố của **category\_id** theo từng **publish\_country** (quốc gia xuất bản).

A chart of different colors

Description automatically generated

Biểu đồ **Violin Chart** giúp bạn hiểu rõ hơn về phân phối của dữ liệu **category\_id** cho từng **publish\_country**. Mỗi "hình vĩ cầm" (violin) thể hiện sự phân bố của dữ liệu trong các quốc gia xuất bản khác nhau. Cấu trúc của biểu đồ bao gồm:

* **Đoạn thân**: Đường viền thể hiện mật độ phân phối của dữ liệu.
* **Vùng rộng nhất**: Cho biết phần lớn dữ liệu tập trung ở đâu.
* **Thanh ngang (tương tự boxplot)**: Thể hiện các phần trăm phân vị (quartiles) và điểm trung vị.

## Density chart

Biểu đồ **Density Chart** là một dạng biểu đồ giúp trực quan hóa phân phối xác suất của một biến liên tục. Nó sử dụng đường cong mật độ thay vì histogram để biểu diễn sự phân bố của dữ liệu. Trong trường hợp này, biểu đồ mật độ được vẽ cho **category\_id** của các video, phân theo các **publish\_country** (quốc gia xuất bản).

A graph of different colored lines

Description automatically generated

Biểu đồ **Density Chart** giúp bạn hiểu rõ hơn về phân phối của **category\_id** trong các **publish\_country** khác nhau. Mỗi đường biểu diễn mật độ phân phối của category\_id cho một quốc gia xuất bản cụ thể. Các điểm sau có thể rút ra từ biểu đồ này:

* **Độ dốc của đường mật độ**: Đường có độ dốc lớn cho thấy sự phân bố của category\_id có xu hướng tập trung vào một giá trị nhất định.
* **Khu vực dưới đường mật độ**: Khu vực này cho thấy xác suất của category\_id tại các mức giá trị khác nhau.
* **Sự chồng lắp của các đường**: Nếu các đường mật độ chồng lên nhau, điều này có thể chỉ ra sự tương đồng về phân phối giữa các quốc gia trong category\_id.

## Boxplot chart

Biểu đồ **Boxplot** (hay còn gọi là biểu đồ hộp và râu) giúp trực quan hóa sự phân phối của một biến liên tục thông qua các thông số thống kê như giá trị trung vị, các phần tử bách phân vị (quartiles) và các giá trị ngoại lai (outliers). Biểu đồ này thường được sử dụng để so sánh phân phối của nhiều nhóm dữ liệu.

A chart with different colored squares

Description automatically generated with medium confidence

Biểu đồ **Boxplot** giúp phân tích sự phân phối của **category\_id** cho từng quốc gia (theo publish\_country) và sắp xếp các quốc gia theo giá trị trung vị của category\_id. Các yếu tố quan trọng trong Boxplot bao gồm:

* **Phần hộp (box)**: Đại diện cho phạm vi giữa các phần tử bách phân vị 1 (Q1) và phần tử bách phân vị 3 (Q3), bao gồm 50% dữ liệu.
* **Đường giữa hộp (median line)**: Đại diện cho giá trị trung vị (median) của dữ liệu.
* **Các râu (whiskers)**: Biểu thị các giá trị ngoài phạm vi của hộp (từ Q1 đến Q3), nhưng không phải là các giá trị ngoại lai.
* **Các điểm ngoại lai (outliers)**: Các điểm nằm ngoài các râu của hộp, cho thấy các giá trị cực trị.

## Ridgeline plot

Biểu đồ **Ridgeline** là một kiểu biểu đồ dùng để mô tả sự phân phối của dữ liệu qua nhiều nhóm hoặc phân loại khác nhau. Biểu đồ này hiển thị các hàm mật độ xác suất (probability density functions) của mỗi nhóm dữ liệu, tạo thành một dạng "đỉnh núi" với các đường cong chồng lên nhau.

A graph of data showing a distribution of views

Description automatically generated with medium confidence

Biểu đồ **Ridgeline** giúp so sánh sự phân phối của category\_id trong từng ngày trong tuần (published\_day\_of\_week). Mỗi đường cong đại diện cho sự phân phối của category\_id trong một ngày cụ thể, và các đường cong này được xếp chồng lên nhau để tạo thành các đỉnh, giống như các dãy núi.

## Correlation plot

Biểu đồ **Correlation Plot** là một công cụ trực quan hóa mạnh mẽ giúp hiển thị mối quan hệ giữa các biến số trong một tập dữ liệu. Mỗi biến trong tập dữ liệu được so sánh với tất cả các biến còn lại thông qua các biểu đồ scatter plot và biểu đồ histogram. Biểu đồ này giúp phát hiện các mối quan hệ tuyến tính hoặc phi tuyến tính giữa các biến, và có thể cung cấp những thông tin quan trọng về dữ liệu.

A screenshot of a graph

Description automatically generated

Đồ thị dựa trên dữ liệu tổng hợp

## Stacked plot

Biểu đồ **Stacked Barplot** là một loại biểu đồ cột, trong đó mỗi cột đại diện cho một nhóm dữ liệu, nhưng thay vì chỉ một giá trị, mỗi cột được chia thành các phần nhỏ đại diện cho các phân nhóm trong nhóm đó. Biểu đồ này rất hữu ích khi muốn so sánh tổng thể các nhóm dữ liệu đồng thời phân tích sự phân bổ của các phân nhóm trong mỗi nhóm.

A graph of different colors

Description automatically generated

## Circular barplot

**Circular Barplot** là một dạng biểu đồ thanh (barplot) được vẽ theo vòng tròn, giúp thể hiện dữ liệu dạng phân loại hoặc liên tục theo cách thức thú vị và dễ tiếp cận. Thay vì các thanh thẳng đứng như trong biểu đồ cột truyền thống, các thanh trong biểu đồ vòng được vẽ theo dạng vòng cung, cung cấp một cái nhìn khác biệt và có thể dễ dàng phân biệt các phân nhóm.

A circular bar plot with numbers and a rainbow colored circle

Description automatically generated with medium confidence

## Heatmap

**Heatmap** là một công cụ trực quan hóa dữ liệu giúp thể hiện mối quan hệ giữa các biến số theo một cách sinh động thông qua màu sắc. Màu sắc trong heatmap sẽ thể hiện giá trị của các biến, giúp người xem dễ dàng nhận biết sự phân bố và mức độ của các giá trị này trong không gian hai chiều.

A graph of a heatmap

Description automatically generated

## Area chart

**Area Chart** là một loại biểu đồ trực quan hóa dữ liệu có thể giúp bạn hiểu sự thay đổi của một hoặc nhiều tập hợp dữ liệu theo thời gian. Biểu đồ này kết hợp giữa đường (line chart) và các vùng (area), giúp thể hiện được mối quan hệ giữa các giá trị và sự biến động của chúng trong suốt các khoảng thời gian hoặc các phân loại khác.

A graph with numbers and lines

Description automatically generated

## Bubble chart

**Bubble Chart** là một dạng biểu đồ scatter (biểu đồ phân tán) nhưng có sự bổ sung các "bong bóng" với kích thước thay đổi theo giá trị của một biến số. Nó được sử dụng để thể hiện mối quan hệ giữa ba biến số, trong đó hai biến được biểu thị trên trục x và y, trong khi biến còn lại được thể hiện qua kích thước của các bong bóng. Ngoài ra, mỗi bong bóng có thể được tô màu khác nhau để phân biệt các nhóm dữ liệu.

A graph with dots and numbers

Description automatically generated

KẾT LUẬN

Em xin gửi lời biết ơn sâu sắc đến thầy Lê Văn Hạnh đã dành nhiều thời gian và tâm huyết hướng dẫn nghiên cứu và giúp em hoàn thành môn học.

Em đã có nhiều cố gắng hoàn thiện dự án bằng tất cả năng lực của mình, tuy nhiên không thể tránh khỏi nhiều thiếu sót, rất mong nhận được những đóng góp quý báu của quý thầy cô và các bạn.