CHƯƠNG 3

LÀM SẠCH VÀ TIỀN XỬ LÝ DỮ LIỆU

Giảng viên: Nguyễn Anh Thư

Khoa: Khoa học ứng dụng

NỘI DUNG BÀI HỌC

3.1 THĂM DÒ DỮ LIỆU

- 3.1.1. Phân tích đơn biến – Univariate Analysis
- 3.1.2. Phân tích đa biến – Multivariate Analysis
- 3.1.3. Distribution Fitting

3.2 LÀM SẠCH DỮ LIỆU

- 3.2.1. Biến đổi dữ liệu
- 3.2.2. Xử lý dữ liệu thiếu và các ngoại lệ
- 3.2.3. Phát hiện và loại bỏ trùng lặp

3.3 TIỀN XỬ LÝ DỮ LIỆU

MỤC TIÊU BÀI HỌC

Sau khi học xong bài này, cần nắm được các vấn đề sau:

- Các phương pháp phân tích đơn biến, đa biến và kiểm tra phân phối phù hợp với dữ liệu.
- Các kỹ thuật cơ bản trong việc làm sạch dữ liệu.
- Sau khi khám phá được nội hàm bên trong dữ liệu, ta thực hiện tiền xử lý dữ liệu để chuẩn bị đưa vào khai thác sâu hơn (như xây dựng mô hình,…).

3.1 THĂM DÒ DỮ LIỆU

- Khi làm việc với một tập dữ liệu, bước đầu tiên là hiểu ý nghĩa và đặc điểm chính của nó, tức là các sự kiện hoặc đối tượng mà dữ liệu đề cập đến và các thuộc tính mà nó biểu thị, cần phải hiểu được nội hàm. Mục tiêu của việc thăm dò trên một tập dữ liệu bao gồm:
  - Xác định kiểu loại của tập dữ liệu (có cấu trúc, phi cấu trúc, bán cấu trúc).
  - Nếu dữ liệu không phải phi cấu trúc, cần hiểu được schema của nó (các thuộc tính cấu thành và cách tổ chức dữ liệu).
  - Xác định kiểu miền của từng thuộc tính (danh mục, thứ tự, hay số) và đặc điểm của giá trị trong miền đó (giá trị điển hình, phạm vi giá trị, v.v.).

3.1 THĂM DÒ DỮ LIỆU

- Mục tiêu trong phần này là làm việc với dữ liệu có cấu trúc và lược đồ đã biết.
- Các bước:
  1. Phân tích từng thuộc tính riêng lẻ được gọi là phân tích đơn biến (univariate analysis) trong thống kê, tập trung vào một biến tại một thời điểm.
  2. Phân tích các mối quan hệ giữa các thuộc tính được gọi là phân tích đa biến trong thống kê. Thông thường một thuộc tính trong dữ liệu thường có liên quan đến các thuộc tính khác vì mỗi thuộc tính mô tả một khía cạnh/ đặc điểm của đối tượng, sự kiện nên nhiều khi chúng thể hiện một số đặc điểm chung.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

- Cần xác định loại dữ liệu của thuộc tính (domain type) như categorical (danh mục), numerical (số), hoặc ordinal (thứ tự). Ngoài ra, cần xem xét liệu kiểu dữ liệu được gán cho thuộc tính (ví dụ, string, number, date) có phù hợp với loại dữ liệu của nó không.
- Ví dụ:
  - Các categorical có thể được mã hóa bằng số ('1', '2', ...), nhưng không có ý nghĩa số học hoặc thứ tự.
  - Thông tin thời gian (temporal data) thường bị nhập dưới dạng chuỗi thay vì kiểu dữ liệu ngày/thời gian, điều này gây khó khăn cho việc phân tích.
  - Thuộc tính dạng số không nên được lưu dưới dạng chuỗi (string), hoặc ngược lại, ngày tháng không nên lưu dưới dạng số.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

- Thường đánh giá về:
  - Khoảng giá trị (min, max).
  - Xu hướng trung tâm (mean, median, mode).
  - Độ phân tán (độ lệch chuẩn, phân vị,…).

➔ Với các thước đo này ta sử dụng các hàm AGGREGATE FUNCTION.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình - MEAN**

```
SELECT Avg(Attr) as mean
FROM Data;
```

≡

```
SELECT Sum(Attr) /
(Count(Attr) * 1.0) as mean
FROM Data;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình - MEAN**

VD1: Từ bảng StudentScores(student_id, student_name, score, exam_date, subjects_enrolled) như dưới đây, hãy tính điểm trung bình của các sinh viên.

student_id	student_name	score	exam_date	subjects_enrolled
1	Nguyen Van A	8.5	2024-01-15	3
2	Tran Thi B	9	2024-01-15	1
3	Le Van C	7	2024-01-15	2
4	Pham Thi D	8	2024-01-16	5
5	Hoang Van E	7.5	2024-01-16	4

student_id	student_name	score	exam_date	subjects_enrolled
6	Dang Thi F	6.5	2024-01-16	2
7	Vo Van G	9.5	2024-01-17	1
8	Bui Thi H	8	2024-01-17	3
9	Do Van I	7.5	2024-01-17	3
10	Hoang Bach K	7.2	2024-01-16	5

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình - MEAN**

VD1: Từ bảng StudentScores(student_id, student_name, score, exam_date, subjects_enrolled) như dưới đây, hãy tính điểm trung bình của các sinh viên.

Cách 1:

```
SELECT Avg(score) as mean
FROM StudentScores;
```

Cách 2:

```
SELECT Sum(score) / (Count(score) * 1.0) as mean
FROM StudentScores;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình - MEAN**

- Trong xác suất thống kê, nếu một giá trị x_i xuất hiện f_i lần trong tổng số N quan sát.
- Khi đó, xác suất thực nghiệm là:

$$P(x_i) \approx \frac{f_i}{N} = \frac{f_i}{\sum f_i} \qquad (1)$$

$$\sum x_i \cdot P(x_i) = \sum \left(x_i \cdot \frac{f_i}{\sum f_i} \right) \qquad (2)$$

➔ (2) là công thức TB KỲ VỌNG $\mathrm{E(X)} = \sum x_i \cdot P(x_i)$

3.1 THĂM DÒ DỮ LIỆU

❖ **Giá Trị Trung Bình –** Sử dụng xác suất

```
WITH NHistogram(Value, Prob) AS
(       SELECT Attr, sum(1.0/total)
        FROM  Data,  (SELECT  COUNT(*)  AS
                total FROM Data) AS Temp
        GROUP BY Attr
)
SELECT Sum(Value * Prob) AS mean
FROM NHistogram;
```

❖ **Giá Trị Trung Bình –** Sử dụng tần số

```
WITH Histogram(Value, Frequency) AS
(       SELECT Attr, count(*)
        FROM Data
        GROUP BY Attr
)
SELECT (1.0 * Sum(Value * Frequency)) /
Sum(Frequency) AS mean
FROM Histogram;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ Giá Trị Trung Bình - MEAN

VD2: Từ bảng StudentScores(student_id, student_name, score, exam_date, subjects_enrolled) như dưới đây, tính số môn học trung bình mà các sinh viên tham gia (**sử dụng 2 cách xác suất và tần số**, biết số môn học tối thiểu mà sinh viên tham gia nằm trong khoảng từ 1 đến 4).

student_id	student_name	score	exam_date	subjects_enrolled
1	Nguyen Van A	8.5	2024-01-15	3
2	Tran Thi B	9	2024-01-15	1
3	Le Van C	7	2024-01-15	2
4	Pham Thi D	8	2024-01-16	5
5	Hoang Van E	7.5	2024-01-16	4

student_id	student_name	score	exam_date	subjects_enrolled
6	Dang Thi F	6.5	2024-01-16	2
7	Vo Van G	9.5	2024-01-17	1
8	Bui Thi H	8	2024-01-17	3
9	Do Van I	7.5	2024-01-17	3
10	Hoang Bach K	7.2	2024-01-16	5

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình - MEAN**

VD2: *Cách 1:*

```
WITH NHistogram(Value, Prob)
AS ( SELECT Subjects_Enrolled, sum(1.0/total)
     FROM StudentScores, (SELECT COUNT(*) AS total FROM StudentScores) AS Temp
     GROUP BY Subjects_Enrolled )
SELECT Sum(Value * Prob) AS mean FROM NHistogram;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình - MEAN**

VD2: *Cách 2:*

```
WITH Histogram(Value, Frequency)
AS ( SELECT Subjects_Enrolled, count(*)
     FROM StudentScores GROUP BY Subjects_Enrolled )
SELECT (1.0 * Sum(Value * Frequency)) / Sum(Frequency) AS mean
FROM StudentScores;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – TRIMMED MEAN**

- GTTB rất nhạy bởi các giá trị ngoại lai (outlier) và các giá trị cực trị.
- Ví dụ nếu có một giá trị rất cao hoặc rất thấp thì nó sẽ ảnh hưởng rất lớn đến giá trị trung bình.

➔ Sử dụng TRIMMED MEAN – giá trị trung bình được cắt tỉa: GTTB sẽ được tính toán sau khi bỏ qua các giá trị cực trị, thường là GTNN và GTLN, hoặc khái quát hơn là loại bỏ k% giá trị cao nhất/ thấp nhất.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – TRIMMED MEAN**

Loại bỏ các giá trị cực trị là GTLN và GTNN

```
SELECT avg(A)
FROM Data,
    (SELECT max(A) AS Amax FROM Data) AS T1,
    (SELECT min(A) AS Amin FROM Data) AS T2
WHERE A < Amax and A > Amin;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – TRIMMED MEAN**

Loại bỏ các giá trị cực trị là GTLN và GTNN

```
SELECT AVG(score)
FROM   studentscores,
       (SELECT min(score) As min_score FROM
              studentscores) AS T1,
       (SELECT max(score) AS max_score FROM
              studentscores) as T2
WHERE min_score < score AND score < max_score;
```

avg(score)
7.837499976158142

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – TRIMMED MEAN**

Loại bỏ k% GTLN và GTNN:

Bước 1. Sắp xếp dữ liệu theo chiều tăng/giảm dần giá trị:

```
WITH RankedData
AS (SELECT A,
    ROW_NUMBER() OVER (ORDER BY A ASC) AS rank_asc,
    ROW_NUMBER() OVER (ORDER BY A DESC) AS rank_desc
    FROM Data)
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – TRIMMED MEAN**

Loại bỏ k% GTLN và GTNN:

Bước 2. Loại bỏ k% phần tử là GTLN và GTNN:

```
WITH TrimmedData
AS (SELECT A
    FROM RankedData
    WHERE rank_asc > (SELECT COUNT(*) * 0.01 * k FROM Data)
    AND rank_desc > (SELECT COUNT(*) * 0.01 * k FROM Data)
)
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – TRIMMED MEAN**

Loại bỏ k% GTLN và GTNN:

Bước 3. Tính TRIMMED MEAN:

```
SELECT AVG(A) AS TrimmedMean
FROM TrimmedData;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ Giá Trị Trung Bình – GEOMETRIC MEAN

- Geometric mean (trung bình nhân) có công thức tổng quát như sau: $\sqrt[n]{x_1 * x_2 * \cdots * x_n}$.
- Ưu điểm: ít nhạy cảm với các giá trị ngoại lại so với trung bình cộng, đặc biệt với các giá trị lớn.
- Nhược điểm: vẫn bị ảnh hưởng bởi các giá trị nhỏ.
- SQL không có hàm AGGREGATE để tính trực tiếp.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – GEOMETRIC MEAN** $\sqrt[n]{x_1 * x_2 * \cdots * x_n}$.

- **Cách 1**: giữ nguyên công thức

Bước 1: Sử dụng hàm logarit để tính tích:

$$\boldsymbol{log(a * b) = log\, a + log\, b}$$

$$\Rightarrow \boldsymbol{a * b = \exp(\log a + \log b)}$$

```
SELECT exp(sum(log(Attr))
FROM Data;
```

Bước 2: Sử dụng hàm POW trong SQL với số mũ là $\frac{1}{n}$ để tính căn.

```
SELECT pow(exp(sum(log(Attr)), 1.0/total)
FROM Data, (SELECT count(Attr) AS total FROM Data);
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Giá Trị Trung Bình – GEOMETRIC MEAN** $\sqrt[n]{x_1 * x_2 * \cdots * x_n}$.

- **Cách 2**: biến đổi thành công thức tương đương $\sqrt[n]{\boldsymbol{x_1 * x_2 * \cdots * x_n}} = \boldsymbol{exp}\,\frac{\boldsymbol{ln\, x_1 + ln\, x_2 + \cdots + ln\, x_n}}{\boldsymbol{n}}$.

```
SELECT exp(sum(log(Attr)) / count(Attr))
FROM Data;
```

Vì sum/count = average nên câu lệnh trên có thể thay thế thành:

```
SELECT exp(avg(log(Attr))
FROM Data;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ Giá Trị Trung Bình – GEOMETRIC MEAN

- Khi nào thì nên sử dụng trung bình nhân?
  - Hữu ích khi tính toán tỷ lệ lãi suất trong ngân hàng.
  - Đo lường tốc độ tăng trưởng/ suy giảm trung bình trong tài chính, kinh tế và sinh học.
  - Phù hợp với các tường hợp có tỷ lệ thay đổi mạnh hoặc dữ liệu có biến động lớn.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ MODE

- Là giá trị xuất hiện nhiều nhất trong dữ liệu (có thể có nhiều mode).
- Trong SQL không có hàm tính mode mà phải tính toán theo các bước như sau:

```
WITH Histogram AS
 (SELECT Value AS val, count(*) AS freq FROM Data GROUP BY Attr)
SELECT val
FROM Histogram, (SELECT max(freq) AS top FROM Histogram) AS T
WHERE freq = top;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ MODE

- **VD3**: Trong các chuyến bay đến NY, hãy tính điểm đến (destination) được ghé thăm nhiều nhất (điểm đến với nhiều chuyến bay đến đó).

FlightID	Origin	Destination	FlightDate
1	NY	Los Angeles	2024-02-01
2	NY	Chicago	2024-02-02
3	NY	Los Angeles	2024-02-03
4	NY	San Francisco	2024-02-04
5	NY	Chicago	2024-02-05

FlightID	Origin	Destination	FlightDate
6	NY	Los Angeles	2024-02-06
7	NY	Miami	2024-02-07
8	NY	San Francisco	2024-02-08
9	NY	Los Angeles	2024-02-09
10	NY	Chicago	2024-02-10

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ MODE

- **VD3**: Trong các chuyến bay đến NY, hãy tính điểm đến (destination) được ghé thăm nhiều nhất (điểm đến với nhiều chuyến bay đến đó).

  Cách 1:

```
SELECT Destination, COUNT(*) AS FlightCount
FROM Flights
GROUP BY Destination
ORDER BY FlightCount DESC
LIMIT 1;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ MODE

- **VD3**: Trong các chuyến bay đến NY, hãy tính điểm đến (destination) được ghé thăm nhiều nhất (điểm đến với nhiều chuyến bay đến đó).

Cách 2:

```
WITH Histogram AS  (SELECT Destination AS val, count(*) AS freq
                    FROM Flights GROUP BY Destination)
SELECT val, COUNT(*) AS FlightCount
FROM Histogram, (SELECT max(freq) AS top FROM Histogram) AS T
WHERE freq = top;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ MEDIAN

- Là giá trị xuất hiện ở vị trí giữa khi các giá trị được sắp xếp theo thứ tự nhỏ đến lớn sao cho số lượng số bên trái và phải nó là bằng nhau, được gọi là trung vị.
  - ➢ Nếu số lượng phần tử là lẻ, trung vị là giá trị ở giữa.
  - ➢ Nếu số lượng phần tử là chẵn, trung vị là trung bình của hai giá trị ở giữa.
- Trung vị ít bị ảnh hưởng bởi các giá trị ngoại lai hơn so với trung bình cộng.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ MEDIAN

- Tính trung vị trong SQL khá phức tạp do cần sắp xếp các giá trị (ORDER BY) và xác định vị trí ở giữa, thường sử dụng kết hợp LIMIT và OFFSET để lấy giá trị mong muốn:
  - Xác định tổng số phần tử `size`.
  - Sắp xếp dữ liệu sử dụng `ORDER BY`.
  - `MOD(size, 2)` kiểm tra tổng số phần tử chẵn hay lẻ .
  - Xác định vị trí bắt đầu của trung vị bằng cách
    - Xác định vị trí chính giữa của DL `CEIL(size / 2.0)`
    - `OFFSET` dịch con trỏ đến vị trí chính giữa của DL.
    - Tính trung bình của các phần tử chính giữa.

```
SELECT avg(Attr)
FROM (SELECT Attr FROM Data,
      (SELECT count(*) as size FROM Data)
      ORDER BY value
      LIMIT 2 - MOD(size, 2)
      OFFSET CEIL(size / 2.0) ) AS T;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ **Mức Độ Phân Tán**

- Đơn giản nhất là khoảng giá trị (range) giữa GTLN và GTNN, đã có sẵn trong SQL thông qua hàm MAX() và MIN().
- Độ lệch chuẩn (standard deviation) thường sử dụng phổ biến để đo độ phân tán, được tích hợp sẵn trong hệ quản trị CSDL thông qua hàm STD(). Công thức :

$$\sqrt{\frac{(\Sigma_{i=1}^{n} x_i^2)}{(n-1)} - \left(\frac{(\Sigma_{i=1}^{n} x_i)}{(n-1)}\right)^2}$$

- Phương sai (variance) cũng cần để phân tích trong một số trường hợp, được xác định là bình phương của độ lệch chuẩn, là một phép toán được tích hợp sẵn trong hầu hết các hệ thống thông qua hàm VARIANCE().

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ SKEWNESS VÀ KURTOSIS

- Skewness (Độ nghiêng/ Độ chệch) được sử dụng để đo lường sự đối xứng của phân phối.
- Phân phối đối xứng khi (mean) bằng giá trị trung vị (median), với độ nghiêng bằng 0, trong đó phân phối lệch phải (skew dương) là phân phối có đuôi kéo dài về bên phải, phân phối lệch trái (skew âm) có đuôi kéo dài về bên trái. Độ chệch này cho biết phân phối có cân bằng xung quanh GTTB hay không.

$$Skewness = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right)^{\frac{3}{2}}}$$

- n là số lượng giá trị,
- x_i là giá trị từng biến và $\overline{x}$ là GTTB.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ SKEWNESS VÀ KURTOSIS

- Kurtosis (độ nhọn) sử dụng để đo lường mức độ tập trung của các giá trị xa TB (đuôi phân phối), giúp xác định khả năng xuất hiện, xu hướng phân phối tạo ra các giá trị outlier.

$$Kurtosis = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right)^2}$$

- n là số lượng giá trị,
- x_i là giá trị từng biến và $\overline{x}$ là GTTB.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

a. VỚI CÁC THUỘC TÍNH KIỂU SỐ

❖ SKEWNESS VÀ KURTOSIS

- Kurtosis (độ nhọn) sử dụng để đo lường mức độ tập trung của các giá trị xa TB (đuôi phân phối), giúp xác định khả năng xuất hiện, xu hướng phân phối tạo ra các giá trị outlier.
- Kurtosis thường được so sánh với 3 (kurtosis chuẩn của phân phối chuẩn).
  - ➢ Kurtosis > 3: phân phối có đuôi dày hơn, dễ xuất hiện giá trị ngoại lai (leptokurtic).
  - ➢ Kurtosis = 3: phân phối chuẩn (mesokurtic).
  - ➢ Kurtosis < 3: đuôi mỏng hơn, ít ngoại lai hơn (platykurtic)

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- Khi làm việc với thuộc tính dạng phân loại, công cụ quan trọng nhất trong trường hợp này là histogram – biểu đồ tần suất. Ở dạng đơn giản nhất của histogram, một tập các giá trị trong tập dữ liệu đi kèm với tần suất xuất hiện của chúng, đây là cách biểu diễn đơn giản và trực quan để hiểu sự phân bố của các giá trị thuộc tính phân loại.

- Một cách đơn giản để xây dựng một histogram của một thuộc tính phân loại Attr như sau:

```
SELECT Attr, count(*)

FROM table

GROUP BY Attr;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Binning** là một trong những kỹ thuật tổng quát hơn histogram:
  - Các giá trị của một biến được chia thành các khoảng rời rạc gọi là bin/ bucket, các giá trị rơi vào cùng một khoảng sẽ được thay thế bằng một giá trị đại diện (representative value)
  - Với các biến liên tục: các giá trị được nhóm thành các khoảng và thường là các khoảng giá trị liên tiếp.
  - Với biến phân loại: mỗi giá trị phân loại sẽ ứng với một bin riêng biệt, (có thể nhóm nhiều giá trị phân loại vào cùng một khoảng nếu cần).

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Binning** là một trong những kỹ thuật tổng quát hơn histogram:

Histogram	Binning
Tần suất xuất hiện đại diện cho một bin.	Một giá trị thống kê (GTTB, tổng,…) đại diện cho một bin.

- Binning linh hoạt và hữu ích để xử lý và tổng hợp dữ liệu, đặc biệt là khi làm việc với các biến liên tục hay cần đơn giản hóa dữ liệu.
- Histogram là một dạng cụ thể của binning, trong đó giá trị đại diện là tần suất xuất hiện.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- Để thực hiện binning tổng quát trong SQL, các giá trị cần được chia thành các khoảng và số lượng khoảng cần tạo. Có 2 phương pháp chính:

  - **Equi-depth histogram:** giới hạn các bin sao cho mỗi bin chứa số lượng điểm dữ liệu bằng nhau
  - **Equi-space histogram:** tất cả các bin có cùng độ rộng (cố định độ rộng). Với các thuộc tính phân loại, mỗi khoảng sẽ chứa cùng số lượng giá trị.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Equi-depth histogram:** xác định các phân vị (quantiles) của dữ liệu, là các điểm cắt chia miền giá trị thành các khoảng có cùng số điểm dữ liệu:

  - **Percentiles**: Chia miền giá trị thành 100 khoảng, mỗi khoảng chứa 1% dữ liệu.
  - **Quartiles**: Chia miền giá trị thành 4 khoảng, tương ứng với các mức 25%, 50%, 75%, và 100% dữ liệu.
  - **Deciles**: Chia miền giá trị thành 10 khoảng, tương ứng với các mức 10%, 20%, ..., và 100% dữ liệu.
  - **Median** (trung vị) cũng được xem là một dạng 2-quantile, vì nó chia dữ liệu thành 2 khoảng có số lượng điểm bằng nhau.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Equi-depth histogram:**

VD5: giả sử ta có bảng Heights(age, size) với age là số nguyên, hãy tạo histogram cho thuộc tính age dựa trên quartiles, nghĩa là bin đầu tiên đại diện cho 25% lứa tuổi đầu tiên, bin thứ hai đại diện cho 25-50% lứa tuổi tiếp theo, bin thứ 3 từ 50-75% và bin cuối trên 75% lứa tuổi còn lại.

Gợi ý: đầu tiên ta cần xem có bao nhiêu phần tử và xác định 4 nhóm. Sau đó chia tất cả các phần tử này vào 4 nhóm sau khi đã sắp xếp chúng.

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Equi-depth histogram:**

VD5:

```
WITH OrderedData AS (
    SELECT *, NTILE(4) OVER (ORDER BY age)
    AS quartile FROM Heights )
SELECT
    CASE
        WHEN quartile = 1 THEN 'bottom25'
        WHEN quartile = 2 THEN '25to50'
        WHEN quartile = 3 THEN '50to75'
        WHEN quartile = 4 THEN 'top25'
    END AS quartile, age, size
FROM OrderedData
ORDER BY quartile, age;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Equi-space histogram:** Với thuộc tính phân loại, mỗi khoảng sẽ chứa cùng số lượng giá trị. Phương pháp này thường chỉ áp dụng cho miền dữ liệu thứ tự và dạng số. Độ rộng của các khoảng sẽ quyết định số lượng bin với một miền dữ liệu số nhất định (D). Nếu h là độ rộng cố định, thì số lượng bin sẽ được xác định bởi công thức:

$$number\ of\ bins = \left\lceil \frac{\max(D) - \min(D)}{h} \right\rceil$$

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Equi-space histogram:**

VD6: bảng Heights(age, size) với age là số nguyên, hãy tạo histogram cho thuộc tính age trong đó mỗi khoảng có độ rộng cố định là 4.

```
SELECT age, size, ceil(((age-minage)+1)/4) AS bin
FROM Heights, (SELECT min(age) AS minage FROM Heights) AS T
ORDER BY bin;
```

→ Ánh xạ mỗi giá trị vào một bin bắt đầu từ bin 1, tuổi nhỏ nhất được ánh xạ đến bin 1 bởi (age-minage)+1), nhỏ thứ hai đến bin 2, …

3.1 THĂM DÒ DỮ LIỆU

3.1.1 PHÂN TÍCH ĐƠN BIẾN

b. VỚI CÁC THUỘC TÍNH KIỂU PHÂN LOẠI

- **Equi-space histogram**:
  - Phân chia các bin quá rộng → một vài bin sẽ ẩn đi các đặc điểm quan trọng của dữ liệu.
  - Phân chia các bin quá hẹp → số lượng bin quá lớn, làm cho dữ liệu khá bất thường nếu nó không phù hợp với bất kỳ phân phối nào đã biết.
- Một số quy tắc có thể sử dụng để chọn số lượng khoảng như sau:
  - Nếu có n điểm DL, chọn $\sqrt{n}$ khoảng cho histogram.
  - Quy tắc Sturges: nếu có n điểm DL , chọn số lượng bin là $\lceil \log_2 n + 1 \rceil$.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

- Giả sử A và B là hai thuộc tính trong một bảng và một trong hai cột phụ thuộc (bị ảnh hưởng) vào cột còn lại. Trong trường hợp này ta nói rằng có một thuộc tính là độc lập (trong thống kê nó là biến dự đoán) và một thuộc tính phụ thuộc (biến đầu ra trong thống kê).

- Tùy thuộc kiểu dữ liệu của hai thuộc tính mà ta có 3 trường hợp như sau:
  - Cả hai đều là số.
  - Cả hai đều là phân loại.
  - Một phân loại kết hợp với một số.

- Thực tế, ta không biết liệu hai thuộc tính có độc lập hay không → phân tích với một số thử nghiệm đơn giản để xác định có mối liên hệ nào giữa các thuộc tính hay không.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

- Cách đơn giản nhất có thể thử nghiệm được với **mọi loại thuộc tính** đó là sử dụng xác suất của chúng.

- A và B được gọi là độc lập nhau nếu:

$$P(A,B) = P(A)P(B)$$

- Trong SQL ta có thể xác định như sau như sau: giả sử ta có Data(A,B)

  - Xác định P(A):

```
SELECT A, sum(1.0/total) as PrA
FROM Data, (SELECT count(*) as total FROM Data) as T
GROUP BY A;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

- Trong SQL ta có thể xác định như sau như sau: giả sử ta có Data(A,B)
  - Xác định P(B):

```
SELECT B, sum(1.0/total) as PrB
FROM Data, (SELECT count(*) as total FROM Data) as T
GROUP BY B;
```

  - Xác định P(A, B):

```
SELECT A, B, sum(1.0/total) as PrAB
FROM Data, (SELECT count(*) as total FROM Data) as T
GROUP BY A, B;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

- Trong SQL ta có thể xác định như sau như sau: giả sử ta có Data(A,B)
  - Tính toán sự phụ thuộc theo công thức xác suất đồng thời:

```
SELECT sum(PrAB - (PrA * PrB))
FROM (      SELECT A, B, PrA, PrB, PrAB
            FROM ProbA, ProbB, ProbAB
            WHERE ProbA.A = ProbAB.A and ProbB.B = ProbAB.B
    ) AS Probabilities;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

- Trong SQL ta có thể xác định như sau như sau: giả sử ta có Data(A,B)
  - Một cách khác để kiểm tra sự độc lập của các thuộc tính bằng cách sử dụng PMI (Pointwise Mutual Information) được xác định như sau:

$$PMI(A,B) = \log \frac{P(A,B)}{P(A)P(B)}$$

  - Nếu PMI(A, B)=0 thì chúng độc lập với nhau.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Covariance – hiệp phương sai** – là thước đo đơn giản nhất về mối liên hệ có thể có giữa các thuộc tính, nếu Cov(A, B) = 0 thì chúng độc lập với nhau. Hiệp phương sai được xác định bởi:

$$Cov(A,B) = E\left[\left(A - E(A)\right)\left(B - E(B)\right)\right] \qquad (1)$$

với A, B là các thuộc tính, E là trung bình kỳ vọng.

- Một số công thức tính hiệp phương sai tương đương:

$$Cov(A,B) = \frac{\sum_{i=1}^{N}(a_i - \bar{A})(b_i - \bar{B})}{N-1} \qquad (2)$$

$$= \frac{\sum_{i=1}^{N}(a_i b_i)}{N} - \frac{\sum_{i=1}^{N}(a_i)\sum_{i=1}^{N}(b_i)}{N(N-1)} \qquad (3)$$

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Covariance – hiệp phương sai**

VD7: Bảng Data(ID, Num1, Num2) có dữ liệu như sau, hãy tính Cov(Num1, Num2)

ID	Num1	Num2
1	3.5	7.2
2	5.1	8.4
3	7.8	6.9
4	9.0	10.2
5	11.3	4.5
6	13.7	12.8

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Hệ số tương quan Pearson** – là một hiệp phương sai chuẩn hóa có công thức:

$$\rho = \frac{Cov(A,B)}{std(A)std(B)}$$

với A, B là các thuộc tính:

- std(A) là độ lệch chuẩn của A.
- std(B) là độ lệch chuẩn của B.
- $\boldsymbol{\rho} \in [-\mathbf{1}, \mathbf{1}]$ là hệ số tương quan Pearson.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Hệ số tương quan Pearson** – là một hiệp phương sai chuẩn hóa có công thức:

$$\rho = \frac{Cov(A,B)}{std(A)std(B)}$$

  - $|\rho|$ càng lớn thì A và B có sự tương quan mạnh mẽ
  - **ρ<0**: tương quan âm (A và B di chuyển theo hướng ngược chiều nhau).
  - **ρ>0**: có tương quan dương (A, B di chuyển cùng hướng).
  - **ρ=0**: nghĩa là A và B không có sự tương quan tuyến tính (hay độc lập - theo lý thuyết, tuy nhiên nó có thể tương quan theo hàm mũ).
- Trong SQL, hàm **STD()** có sẵn và được sử dụng để tính độ lệch chuẩn.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Covariance – hiệp phương sai**

VD8: Từ bảng Data(ID, Num1, Num2) có dữ liệu như sau, hãy tạo bảng Probabilities (P(A), P(B), P(A,B)) và tính tương quan Pearson với A là Num1, B là Num2 tương ứng.

ID	Num1	Num2
1	3.5	7.2
2	5.1	8.4
3	7.8	6.9
4	9.0	10.2
5	11.3	4.5
6	13.7	12.8

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Với các thuộc tính thứ tự,** sử dụng mối tương quan về thứ hạng – là thước đo mối quan hệ giữa các thứ hạng trên mỗi biến.
- Ý tưởng: so sánh thứ hạng của thuộc tính (vị trí của chúng theo thứ tự) thay vì các giá trị của thuộc tính.

➔ Thứ tự của các giá trị trong một thuộc tính có liên quan đến thứ tự của các giá trị trong thuộc tính khác không?

- Giả sử ta có một bảng dữ liệu (Data) chứa các thuộc tính A và B. Ngoài ra, bảng này còn có hai thuộc tính bổ sung: Arank - Thứ hạng của các giá trị trong thuộc tính A, Brank - Thứ hạng của các giá trị trong thuộc tính B. Cả hai thuộc tính này đều có các giá trị từ 1 đến n, với n là số lượng bản ghi trong tập dữ liệu.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Với các thuộc tính thứ tự,** sử dụng mối tương quan về thứ hạng – **rank correlation**.
  - **Kendall's τ.**
  - **Spearman's ρ.**

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Với các thuộc tính thứ tự,** sử dụng mối tương quan về thứ hạng – **rank correlation**.
  - **Kendall's τ:** so sánh các cặp giá trị (X, Y) trong tập dữ liệu.

$$\tau = \frac{2 \times (\text{số cặp đồng thuận}) \times (\text{số cặp bất đồng})}{n(n-1)}$$

với n là số lượng cặp dữ liệu; $\tau \in [-1, +1]$ trong đó -1 cho thấy sự bất đồng tuyệt đối, +1 là đồng thuận tuyệt đối và 0 là độc lập với nhau.

- ✓ **Đồng thuận**: (x1, y1) và (x2, y2) được coi là đồng thuận nếu thứ hạng của cả x1 và y1 đều lớn/ nhỏ hơn so với thứ hạng của x2 và y2. Ví dụ, nếu rank(x1) > rank(x2) và rank(y1) > rank(y2) hoặc ngược lại.
- ✓ **Bất đồng**: Nếu thứ hạng của một cặp giá trị không đồng thuận, thì chúng được coi là bất đồng nghĩa là nếu rank(x1) < rank(x2) và rank(y1) > rank(y2) hoặc rank(x1) > rank(x2) và rank(y1) < rank(y2).

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Với các thuộc tính thứ tự,** sử dụng mối tương quan về thứ hạng – **rank correlation**.
  - **Kendall's τ:** Vì công thức trên khá tốn kém do sử dụng tích Descartes nên có một công thức tương ứng khác nhưng hiệu quả hơn:

$$\tau = \frac{2}{n(n-1)} \sum_{i} sign(Arank_i - Brank_i)$$

với n là số lượng cặp dữ liệu; $Arank_i$ là rank của phần tử thứ i trong A (tương tự với $Brank_i$), hàm $sign$ được sử dụng để xác định dấu của phép toán trong hàm và thường là hàm có sẵn trong SQL.

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Với các thuộc tính thứ tự,** sử dụng mối tương quan về thứ hạng – **rank correlation**.
  - **Spearman's ρ:** ý tưởng tương tự như Kendall.

$$\rho = 1 - \frac{6\sum_i (Arank_i - Brank_i)^2}{n^3 - n}$$

với n là số lượng cặp dữ liệu; $Arank_i$ là rank của phần tử thứ i trong A (tương tự với $Brank_i$),

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

a. TRƯỜNG HỢP: CẢ HAI THUỘC TÍNH LÀ KIỂU SỐ

- **Với các thuộc tính thứ tự,** sử dụng mối tương quan về thứ hạng – **rank correlation**.

VD9: Từ bảng Data(ID, Num1, Num2) có dữ liệu như sau, hãy tính hệ số rank theo 2 phương pháp trên.

ID	Num1	Num2
1	3.5	7.2
2	5.1	8.4
3	7.8	6.9
4	9.0	10.2
5	11.3	4.5
6	13.7	12.8

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- Có 2 cách để xem sự ảnh hưởng lẫn nhau giữa hai loại thuộc tính này:

➢ Thuộc tính liên tục ảnh hưởng đến thuộc tính phân loại: sử dụng thuộc tính liên tục để dự đoán thuộc tính phân loại ➔ sử dụng các phương pháp phân loại (classification) để dự đoán (sẽ được đề cập trong chương sau). Ví dụ: dự đoán loại khách hàng (khách hàng tiềm năng, khách hàng hiện tại) dựa trên số tiền họ đã chi tiêu (một thuộc tính liên tục).

➢ Thuộc tính phân loại ảnh hưởng đến thuộc tính liên tục: xem xét liệu thuộc tính phân loại có ảnh hưởng đến giá trị của thuộc tính liên tục hay không ➔ phân tích sự khác biệt giữa các GTTB (means) của thuộc tính liên tục, được tạo ra bởi mỗi hạng mục (category) của thuộc tính phân loại bằng cách sử dụng one-way ANOVA.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- **VD10** về one-way ANOVA: bảng Growth(fertilizer, height) với fertilizer là thuộc tính phân loại, cho biết loại phân bón được sử dụng (ví dụ: loại A, loại B, loại C), height là thuộc tính liên tục, cho biết chiều cao của cây sau một thời gian phát triển. Mục tiêu là xem liệu loại phân bón (thuộc tính phân loại) có ảnh hưởng đến chiều cao của cây (thuộc tính liên tục) hay không.

➔ Để thực hiện điều này, chúng ta sẽ so sánh chiều cao trung bình của cây ở các nhóm khác nhau (tức là, các nhóm được bón các loại phân bón khác nhau). One-way ANOVA sẽ cho chúng ta biết liệu có sự khác biệt đáng kể giữa các nhóm này hay không như sau:

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- **VD10** về one-way ANOVA:
  - **Bước 1:** tính GTTB của giá trị liên tục cho từng nhóm/phân loại.

```
CREATE TABLE group-means AS
SELECT fertilizer, avg(height) as group-mean
FROM Growth
GROUP BY fertilizer;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- **VD10** về one-way ANOVA:
  - **Bước 2:** tính GTTB tổng thể của các GTTB từng nhóm

```
SELECT avg(group-mean) as overall-mean
FROM group-means;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- **VD10** về one-way ANOVA:
  - **Bước 3:** tính tổng của khoảng cách bình phương giữa GTTB tổng thể và GTTB từng phân loại rồi chia cho tổng số nhóm trừ 1 (bậc tự do) → sự thay đổi giữa các nhóm.

```
SELECT sum(pow(group-mean - overall-mean, 2))*size/(num-groups- 1) AS between-groups
FROM    (SELECT count(*) AS size FROM Growth),
        (SELECT count(DISTINCT fertilizer) as num-groups FROM Growth),
        group-means;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- **VD10** về one-way ANOVA:
  - **Bước 4:** tính tổng của khoảng cách bình phương giữa GTTB từng nhóm với từng giá trị trong nhóm đó rồi chia cho số lượng nhóm nhân với tổng số điểm dữ liệu trừ 1 → sự thay đổi trong từng nhóm.

```
SELECT sum(pow(height- group-mean, 2)) /

        (num-groups * (size- 1)) AS within-groups

FROM Growth, group-means

WHERE Growth.fertilizer = group-means.fertilizer;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

b. TRƯỜNG HỢP: MỘT THUỘC TÍNH PHÂN LOẠI – MỘT THUỘC TÍNH KIỂU SỐ

- **VD10** về one-way ANOVA:
  - **Bước 5:** tỷ lệ F là sự thay đổi trong mỗi nhóm chia cho sự thay đổi giữa các nhóm. Tỷ lệ này có thể được so sánh với phân phối F (tra bảng) để xác định xem giá trị thu được có ý nghĩa hay không (F-test).

```
SELECT between-groups / within-groups
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

- Sử dụng kỹ thuật histogram đơn giản để so sánh tần suất: bảng tần số (contingency table) hoặc bảng chéo (cross-tabulation).
- Dựa trên tần suất ta tính χ^2 để kiểm tra tính độc lập. Đơn giản nhất là so sánh số lần hai giá trị cùng đồng thời xuất hiện mà người ta quan sát được (observed) so với số lần hai giá trị cùng đồng thời xuất hiện mà người ta mong đợi (expected) nếu các thuộc tính độc lập với nhau ➔ có thể áp dụng cho các thuộc tính thứ tự và số bằng cách đến tần số xuất hiện của chúng.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

- Bảng Tần Số (Contingency Table) hay Bảng Chéo (Cross-tabulation):

➢ Khi cả hai thuộc tính đều là phân loại, chúng ta sử dụng phương pháp histogram cơ bản để so sánh tần số xuất hiện của các giá trị kết hợp.

➢ Bảng tần số (hay bảng chéo) là một bảng hiển thị tần số của các tổ hợp giá trị của hai thuộc tính phân loại. Nó cho thấy số lượng bản ghi có mỗi sự kết hợp cụ thể của các giá trị thuộc hai thuộc tính.

⇨ Cho phép ta xem có bao nhiêu lần mỗi giá trị của một thuộc tính xuất hiện cùng với mỗi giá trị của thuộc tính kia.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

- Kiểm Định Chi-square χ^2 cho Tính Độc Lập:
  - Dựa trên tần số trong bảng tần số, chúng ta có thể tính toán kiểm định chi-square (χ²) để kiểm tra xem liệu hai thuộc tính có độc lập với nhau hay không.
  - Ý tưởng cơ bản của kiểm định chi-square là so sánh tần số quan sát được (số lượng thực tế các lần xuất hiện) với tần số mong đợi (số lượng các lần xuất hiện nếu hai thuộc tính không liên quan đến nhau).
  - Tần Số Mong Đợi: được tính dựa trên giả định rằng hai thuộc tính là độc lập. Nếu hai thuộc tính hoàn toàn độc lập, chúng ta có thể ước tính tần suất mà một tổ hợp giá trị cụ thể sẽ xuất hiện.
    - ⇨ Kiểm định chi-square so sánh tần số quan sát được với tần số mong đợi. Nếu sự khác biệt giữa hai tần số này đủ lớn, chúng ta có thể kết luận rằng hai thuộc tính không độc lập.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

- Về ý tưởng: nếu thuộc tính A có *n* giá trị dương và B có *m* giá trị dương, giả sử c_{ij} là số lần xuất hiện của cặp (dữ liệu i của A và dữ liệu j của B), khi đó giá trị observed (bắt nguồn từ dữ liệu) được xác định:

$$T(X,Y) = \sum_i \sum_j \frac{\left(c_{ij} - E(i,j)\right)^2}{E(i,j)}$$

Trong đó: $E(i,j) = \frac{c_{i\_} c_{\_j}}{c}$ là GT expected so với c_{ij}.

- $c_{i\_}$ là tổng số lần xuất hiện của giá trị *i* trong A.
- $c_{\_j}$ là tổng số lần xuất hiện của giá trị *j* trong B.
- c là tổng số lần xuất hiện của các cặp giá trị.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

- Khi kiểm định về sự độc lập χ^2, dữ liệu thường được biểu diễn dưới dạng một ma trận/ bảng tần số như sau:

	Y_1	Y_m	Tổng số hàng
X_1	c_{11}	c_{1m}	$c_{1\_} = \sum_j c_{1j}$
⋮	...	...	⋮
X_n	c_{n1}	c_{nm}	$c_{n\_} = \sum_j c_{nj}$
Tổng số cột	$c_{\_1} = \sum_i c_{i1}$	$c_{\_m} = \sum_i c_{im}$	

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

- Tính toán các tần số $c_{i\_}$, $c_{\_j}$, c_{ij}, ➔ bảng tần số từ các giá trị này để tính toán thống kê Chi-square.
- Thống kê này được so sánh với phân phối chi-square với bậc tự do $(n-1) \times (m-1)$ để xác định X và Y có độc lập hay không. **Nếu giá trị thống kê chi-square < giá trị tới hạn ➔ X và Y độc lập**.
- Chú ý: Dữ liệu trong bảng mới nên được tổ chức trong một bảng "gọn gàng" (tidy): từ bảng gốc, ta xác định một bảng nguồn “tidy” sao cho có một cột mô tả được các giá trị hàng trong bảng gốc, một cột mô tả cho các giá trị cột trong bảng gốc và một cột đếm số lần kết hợp của từng cặp hàng-cột để tạo ra bảng tần số.

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

VD11: có bảng CUSTOMERS(City, Status, Count) mô tả về trạng thái đăng ký tạp chí của người dân tại nơi họ cư trú. Liệu nơi sống có ảnh hưởng đến tỷ lệ đăng ký không?

CUSTOMERS		
City	Status	Count
Gotham	Active	1462
Gotham	Stopped	794
Metropolis	Active	749
Metropolis	Stopped	385
Smallville	Active	527
Smallville	Stopped	139

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

VD11:

B1: tính $\sum_i c_{i\_}, \sum_j c_{\_j}, \sum_{ij} c_{ij}$ tương ứng với cột "**Status**", "**City**" và "**Count**".

```
SELECT Cities.perCity, Statuses.perStatus, total

FROM ( (SELECT sum(Count SELECT City, sum(Count) as perCity FROM Customers GROUP BY City)
as Cities,

        (SELECT Status, sum(Count) as perStatus FROM Customers GROUP BY Status) as
Statuses,

        (SELECT sum(Count) as total FROM Customers) as Total

WHERE customer.City = Cities.City and customer.Status = Statuses.Status;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

VD11:

B2: tính tỷ lệ stopped/active tổng thể = tổng số lần xuất hiện của stopped/active chia cho tổng số lần của cả stopped và active.

```
SELECT Statuses.perStatus / total
FROM (SELECT Status, sum(Count) as perStatus FROM Customers GROUP BY Status) as Statuses,
    (SELECT sum(Count) as total FROM Customers) as Total
WHERE customer.City = Cities.City and customer.Status = Statuses.Status;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

VD11:

B3: tính giá trị expected theo city = tổng số lần xuất hiện của từng city nhân với tỷ lệ active/stopped tổng thể.

```
SELECT Cities.perCity * (Statuses.perStatus / total) as expected
FROM (SELECT City, sum(Count) as perCity FROM Customers GROUP BY City) as Cities,
    (SELECT Status, sum(Count) as perStatus FROM Customers GROUP BY Status) as Statuses,
    (SELECT sum(Count) as total FROM Customers) as Total
WHERE customer.City = Cities.City and
        customer.Status = Statuses.Status;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.2 PHÂN TÍCH ĐA BIẾN

c. TRƯỜNG HỢP: CẢ HAI LÀ THUỘC TÍNH PHÂN LOẠI

VD11:

B4: tính độ lệch/ sự sai khác giữa giá trị observed và giá trị expected (đối với active và stopped).

```
<biến đổi như trên>

Select sum(Count) – expected …
```

B5: tính chi-square = độ lệch^2 / giá trị expected.

```
<biến đổi như trên>

Select sum(pow(sum(Count) – expected,2)/expected) …
```

3.1 THĂM DÒ DỮ LIỆU

3.1.3 DISTRIBUTION FITTING – KHỚP PHÂN PHỐI

- Là một kỹ thuật trong phân tích dữ liệu để kiểm tra xem liệu dữ liệu số trong tập dữ liệu có tuân theo một phân phối xác suất chuẩn nào đó hay không. Vì dữ liệu thực tế luôn có sai số, nên việc khớp với phân phối chỉ là gần đúng.
- Ý tưởng: tạo ra các giá trị bằng công thức phân phối và so sánh chúng với các giá trị dữ liệu. Nếu sự khác biệt không đáng kể, ta có thể xem dữ liệu đã được tạo ra bởi một quá trình có phân phối cơ bản.
- Để dự đoán một phân phối, có thể bắt đầu bằng cách sử dụng biểu đồ (histograms) và tính toán các đo lường cơ bản về độ tập trung và phân tán của dữ liệu; ngoài ra ta cần phải kiểm tra xem phân phối đã chọn có phù hợp hay không → sử dụng kiểm định χ^2.

3.1 THĂM DÒ DỮ LIỆU

3.1.3 DISTRIBUTION FITTING – KHỚP PHÂN PHỐI

Quy trình chung để thực hiện việc khớp phân phối:

1. Thiết lập dữ liệu dưới dạng phân phối: Tạo một bảng có schema(value, probability). Điều này có thể được thực hiện bằng cách ước tính xác suất từ tỷ lệ phần trăm được lấy từ số lượng dữ liệu thô.
2. Tìm các tham số của phân phối thực nghiệm: Xác định các tham số như giá trị trung bình và độ lệch chuẩn.
3. Thêm một thuộc tính cho ước lượng vào bảng dữ liệu: Tạo một bảng mới có schema (value, probability, estimate). Cột thứ ba này (estimate) sẽ được điền bằng giá trị null.
4. Chọn một phân phối: Sử dụng công thức của phân phối đã chọn, cùng với giá trị trung bình và độ lệch chuẩn của mẫu, để tạo ra một ước tính cho mỗi giá trị và lưu vào thuộc tính " estimate".
5. So sánh estimate thu được với xác suất thực nghiệm: So sánh các giá trị trong cột " estimate" với các xác suất thực nghiệm.

3.1 THĂM DÒ DỮ LIỆU

3.1.3 DISTRIBUTION FITTING – KHỚP PHÂN PHỐI

VD12: Phân phối dữ liệu chuẩn. Một công ty điện thoại ghi lại thời lượng của các cuộc gọi điện thoại. Dữ liệu được chuyển đổi thành bảng DATA với cột số phút, số cuộc gọi quan sát được với những phút đó. Điều này có thể được thực hiện cho mỗi số phút (1,2,3,...) hoặc dưới dạng biểu đồ với khoảng thời gian, sau khi chọn chiều rộng (0 đến 2 phút, 2 đến 4 phút, v.v. cho chiều rộng 2 phút). Ta tính toán ước tính cho μ (trung bình) và σ2 (phương sai) từ dữ liệu trong bảng DATA. Sau đó, ta áp dụng hàm mật độ của phân phối chuẩn:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

cho cột số phút (x) để tạo ra cột số phút ước lượng.

3.1 THĂM DÒ DỮ LIỆU

3.1.3 DISTRIBUTION FITTING – KHỚP PHÂN PHỐI

VD12: Phân phối dữ liệu chuẩn.

Trong SQL có thể làm như sau:

- Tạo bảng DATA và điền dữ liệu vào đó: nếu dữ liệu ở dạng thô Raw_data(call_id, num_minutes) thì truy vấn sử dụng GROUP BY và COUNT để tạo ra bảng với xác xuất thay thế.

```
CREATE TABLE DATA AS

SELECT num_minutes, sum(1.0/ total) as prob

FROM Raw_data, (SELECT count(*) as total FROM Raw_data)

GROUP BY num_minutes;
```

3.1 THĂM DÒ DỮ LIỆU

3.1.3 DISTRIBUTION FITTING – KHỚP PHÂN PHỐI

VD12: Phân phối dữ liệu chuẩn.

Trong SQL có thể làm như sau:

- Tính toán trung bình và độ lệch chuẩn trong dữ liệu. Sau đó, điền vào cột expected frequencies bằng cách áp dụng công thức phân phối chuẩn cho cột dữ liệu và giá trị trung bình và độ lệch chuẩn.

```
UPDATE NORMAL

SET expected-freq = (1 / sqrt(2*pi()*stddev)) *

        exp(- (pow(observed-freq- avg, 2) / (2*stdev)));
```

3.1 THĂM DÒ DỮ LIỆU

3.1.3 DISTRIBUTION FITTING – KHỚP PHÂN PHỐI

VD12: Phân phối dữ liệu chuẩn.

Trong SQL có thể làm như sau:

- So sánh tần số quan sát được (observed frequencies) và tần số dự kiến (expected frequencies) và xem sự khác biệt có đủ nhỏ hay không.

  - Đơn giản hơn: thêm dấu trị tuyệt đối và tính theo độ lệch chuẩn.
  - Phức tạp hơn: kiểm tra chi bình phương.

  ⇨ Phân bố phù hợp có thể được sử dụng để tìm và loại bỏ các outlier.

3.2 LÀM SẠCH DỮ LIỆU

- Làm sạch dữ liệu là tập hợp các hoạt động và kỹ thuật để xác định và loại bỏ các vấn đề với dữ liệu trong một tập dữ liệu.
- Mục tiêu: loại bỏ dữ liệu "xấu" hoặc "sai". Tuy nhiên, việc xác định dữ liệu này có thể rất khó khăn, tùy thuộc vào miền và loại dữ liệu.
- **Đối với các loại dữ liệu liệt kê** (finite enumerated types): Cần một kiểm tra phần tử đơn giản. Ví dụ, tháng trong năm có thể được biểu diễn bằng số nguyên (chỉ các giá trị từ 1 đến 12) hoặc bằng chuỗi ("January", ..., "December"). Việc phân biệt dữ liệu "tốt" và "xấu" thường không khó, trừ khi có sử dụng mã hóa không chuẩn.

3.2 LÀM SẠCH DỮ LIỆU

- **Đối với các miền dựa trên mẫu (pattern-based domains):** (ví dụ: số điện thoại hoặc địa chỉ IP), một kiểm tra có thể loại trừ các giá trị không phù hợp với mẫu. Tuy nhiên, khi một giá trị không tuân theo mẫu dự kiến, có thể là do định dạng sai. Các giá trị này cần được viết lại, không nên bỏ qua hoặc xóa. Việc phân biệt giá trị định dạng sai và giá trị sai hoàn toàn có thể trở nên khó khăn.
- **Đối với các miền mở** (open-ended domains): (như hầu hết các phép đo), việc xác định giá trị "xấu" hoặc "sai" không rõ ràng. Cần kết hợp phân tích sâu các giá trị hiện có và thông tin miền để suy ra phạm vi giá trị có thể và/hoặc có khả năng xảy ra. Ngay cả khi có kiến thức này, việc phân biệt giá trị "tốt" và "xấu" vẫn có thể rất khó.

3.2 LÀM SẠCH DỮ LIỆU

- Một số vấn đề cơ bản có thể giải quyết để làm sạch dữ liệu:
  - **Dữ liệu chính xác**: đảm bảo các giá trị trong dữ liệu đúng loại và đúng định dạng. Cần kiểm tra xem DL đã được đọc chính xác vào CSDL chưa do đôi khi các số có thể bị đọc nhầm thành chuỗi do dấu phẩy, hoặc các ngày tháng không được nhận dạng đúng. Ngay cả khi dữ liệu được đọc đúng, định dạng của nó có thể không phù hợp cho việc phân tích.
  - **Thiếu giá trị dữ liệu**: Trong dữ liệu dạng bảng, giá trị thiếu có nghĩa là các bản ghi bị thiếu một số giá trị thuộc tính. Việc phát hiện giá trị thiếu có thể khó khăn nếu chúng được đánh dấu theo cách không chuẩn. Các chiến lược chung xử lý giá trị thiếu:
    - Bỏ qua (xóa) dữ liệu bị ảnh hưởng (thuộc tính hoặc bản ghi).
    - Dự đoán (điền vào) các giá trị bị thiếu dựa trên các giá trị khác của cùng thuộc tính hoặc các thuộc tính khác.

3.2 LÀM SẠCH DỮ LIỆU

- Một số vấn đề cơ bản có thể giải quyết để làm sạch dữ liệu:
  - **Outlier**: là các giá trị dữ liệu có đặc điểm rất khác biệt so với hầu hết các giá trị dữ liệu khác trong cùng một thuộc tính. Việc phát hiện các giá trị ngoại lệ rất khó khăn vì đây là một khái niệm mơ hồ, phụ thuộc vào ngữ cảnh. Ví dụ: trong một tập dữ liệu về người với thuộc tính chiều cao, khi đo bằng feet, giá trị thường nằm trong khoảng 4.5 đến 6.5. Bất kỳ ai thấp hoặc cao hơn đều được coi là rất thấp hoặc rất cao. Tuy nhiên, chắc chắn có những người trên thế giới rất thấp (và rất cao).
  - **Trùng lặp dữ liệu**: xảy ra khi hai hoặc nhiều bản ghi trong một tập dữ liệu thực tế đề cập đến cùng một thực thể, sự kiện hoặc quan sát trong thế giới thực. Dữ liệu trùng lặp có thể làm sai lệch phân tích, khiến cho các phân tích thống kê bị lệch, hoặc đưa ra các kết quả không chính xác. Nếu hai bản ghi có các giá trị hoàn toàn giống nhau cho tất cả các thuộc tính, việc phát hiện rất dễ dàng nhưng nếu các bản ghi trùng lặp có thể chứa các giá trị tương tự nhưng không hoàn toàn giống nhau, do các lỗi làm tròn, giới hạn độ chính xác trong đo lường, hoặc các lý do khác ➔ việc phát hiện trở nên khó khăn hơn.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

- Là thao tác thay đổi giá trị của một thuộc tính sang một định dạng hoặc phạm vi nhất định → đảm bảo rằng các giá trị của dữ liệu có thể được hiểu bởi các hàm CSDL → dữ liệu có thể được thao tác theo những cách có ý nghĩa. Trong nhiều trường hợp, cần phải biến đổi dữ liệu trước khi thực hiện bất kỳ phân tích nào khác.

- **Đối với các thuộc tính phân loại,** một biến đổi điển hình là chuẩn hóa (normalization/ standardization), là quá trình thay đổi định dạng hoặc phạm vi của giá trị thuộc tính. Ví dụ, so sánh giữa các chuỗi có thể phân biệt giữa các ký tự chữ hoa và chữ thường, điều này có thể tạo ra sự phân biệt là khác nhau, nghĩa là nếu các giá trị như 'Germany' và 'germany' có trong tập dữ liệu, chúng có thể được coi là khác nhau nếu không chuẩn hóa.

- **Đối với các thuộc tính số,** các phép biến đổi điển hình bao gồm chia tỷ lệ (scaling) và chuẩn hóa (normalization). Việc lựa chọn các phép biến đổi này phụ thuộc nhiều vào kiểu dữ liệu.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

a. VỚI DỮ LIỆU DẠNG SỐ

- **Chuẩn hóa:** biến đổi dữ liệu sao cho có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.Ví dụ điển hình của chuẩn hóa là sử dụng *z-score*, trong đó các giá trị mới được biểu thị bằng độ lệch chuẩn so với GTTB.

$$z(v) = \frac{x - \mu}{sdev}$$

trong đó:

- μ là GTTB của biến v.
- x là một giá trị của biến v cần chuẩn hóa.
- $stdv$ là độ lệch chuẩn của v.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

a. VỚI DỮ LIỆU DẠNG SỐ

- **Chuẩn hóa:** biến đổi dữ liệu sao cho có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.Ví dụ điển hình của chuẩn hóa là sử dụng *z-score*, trong đó các giá trị mới được biểu thị bằng độ lệch chuẩn so với GTTB.

$$z(v) = \frac{x - \mu}{sdev}$$

- Ý nghĩa:
  - Đưa các giá trị về cùng một thang đo; cho biết một giá trị cụ thể cách GTTB bao nhiêu lần độ lệch chuẩn.
  - Giá trị z-score > 0 có nghĩa là giá trị đó lớn hơn giá trị trung bình, trong khi giá trị z-score < 0 có nghĩa là giá trị đó nhỏ hơn giá trị trung bình.
  - Z-score = 0 có nghĩa là giá trị đó bằng GTTB.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

a. VỚI DỮ LIỆU DẠNG SỐ

- **Chia tỷ lệ (scaling):** đảm bảo các giá trị nằm trong một phạm vi nhất định. Cụ thể:
  - Biến đổi tuyến tính: chuyển đổi tất cả các giá trị về thành một số nằm trong đoạn [0-1]→ thường sử dụng *max-min scaler.* VD với giá trị x, ta biến đổi chuẩn hóa như sau:

$$x = \frac{x - min}{max - min}$$

  - Biến đổi phi tuyến logistic: cũng chuyển đổi dữ liệu về khoảng (0-1) theo công thức:

$$x = \frac{1}{1 + e^{-x}}$$

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

a. VỚI DỮ LIỆU DẠNG SỐ

- **Chia tỷ lệ (scaling):** đảm bảo các giá trị nằm trong một phạm vi nhất định. So sánh:

Đặc điểm	Biến đổi tuyến tính	Biến đổi logistic
Dạng hàm	Tuyến tính	Phi tuyến (Sigmoid)
Phạm vi giá trị	Có thể giữ nguyên hoặc đưa về [0,1] → tùy thuộc phép biến đổi	Luôn nằm trong khoảng (0,1)
Ảnh hưởng đến phân phối	Không thay đổi, duy trì hình dạng phân phối ban đầu	Làm giảm ảnh hưởng của ngoại lai, nén giá trị lớn/nhỏ, thay đổi hình dạng phân phối
Ứng dụng chính	Chuẩn hóa dữ liệu, biến đổi đơn giản	Hồi quy logistic, mạng nơ-ron, bài toán phân loại

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

a. VỚI DỮ LIỆU DẠNG SỐ

- Trường hợp thuộc tính có phân phối lệch: cần điều chỉnh, sử dụng các phép biến đổi để giảm ảnh hưởng của phần đuôi của phân phối:
  - Biến đổi logarit: sử dụng hàm $\log(x)$ với cơ số 2, e hoặc 10 → giảm sự sai khác giữa giá trị lớn, giúp phân phối đối xứng hơn và giảm ảnh hưởng của giá trị vô vùng ở đuôi.
  - Nhân nghịch đảo: sử dụng hàm nghịch đảo 1/x → lật ngược phân phối và giảm khoảng cách giữa giá trị lớn và nhỏ.
  - Lấy căn bậc hai: hàm $sqrt(v)$ kéo các giá trị lớn lại gần hơn, giúp phân phối bớt lệch.
  - Biến đổi lũy thừa: hàm lũy thừa $pow(x, n)$với n là số mũ, $n < 1$ → giảm độ lệch, $n > 1$ → tăng độ lệch.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Làm sạch dữ liệu dạng chuỗi (thuộc tính phân loại) giúp đảm bảo chỉ có duy nhất một tên (chuỗi) đại diện cho mỗi phân loại.
- **Mục tiêu**: loại bỏ các khác biệt nhỏ không liên quan (chữ hoa/ chữ thường) có thể gây ảnh hưởng đến các tác vụ gộp nhóm (GROUP BY) hoặc tìm kiếm giá trị.
- Vì chuỗi có thể biểu diễn nhiều giá trị khác nhau nên không có quy tắc nhất định về xử lý chuỗi. Tuy nhiên, lý tưởng nhất sẽ là **chuẩn hóa các chuỗi**, **sử dụng một tập hợp các tên chuẩn** để tránh nhầm lẫn.
  - Với các domain đóng như danh sách các tháng trong năm, quy tắc đặt tên theo số/tên gọi.
  - Với các domain mở còn lại: đưa ra một tập hợp các quy ước phải tuân thủ trong quá trình phân tích.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

VD13: bảng student(student_id, …, department) được nhập thủ công tên department như sau:

Student-id	…	Department
1	…	Dept of Economics
2	…	Econ
3	…	Department of Econ

```
UPDATE Student

SET Department = "Economics"

WHERE position('Econ' IN Department) > 0;
```

Nhận xét: nhiều tên khoa dù cùng thể hiện một tên nhưng được trình bày khác nhau → cần được thống nhất thành một tên gọi duy nhất.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm "làm sạch" chuỗi: kết hợp với UPDATE để các giá trị được thống nhất → không bị bỏ sót các giá trị hợp lệ và GROUP được chính xác.

TÊN HÀM	CHỨC NĂNG
TRIM (<position> characters FROM string)	Loại bỏ mọi kí tự trong **characters** (mặc định là khoảng trắng) khỏi chuỗi **string**, bắt đầu tự vị trí **position;** position có thể là: 'leading' (đầu chuỗi), 'trailing' (cuối chuỗi) hoặc 'both' (cả hai đầu - mặc định).
LOWER(string)	Biến mọi ký tự trong **string** thành chữ in thường.
UPPER(string)	Biến mọi ký tự trong **string** thành chữ in hoa.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm "làm sạch" chuỗi: kết hợp với UPDATE để các giá trị được thống nhất → không bị bỏ sót các giá trị hợp lệ và GROUP được chính xác.

TÊN HÀM	VÍ DỤ
TRIM (<position> characters FROM string)	SELECT TRIM(LEADING ' ' FROM ' Hello World ') AS result; SELECT TRIM(BOTH 'x' FROM 'xxxHello Worldxxx') AS result;
LOWER(string)	SELECT LOWER('Hello World') AS result;
UPPER(string)	SELECT UPPER('Hello World') AS result;

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm tìm kiếm phần tử:

TÊN HÀM	CHỨC NĂNG
POSITION(substr IN string)	Trả về một số đại diện cho vị trí mà substr xuất hiện lần đầu tiên trong string (được đánh số từ trái sang phải bắt đầu từ 1, nếu chuỗi không được tìm thấy thì trả về 0). Trong cách 2 có **start_position** để cung cấp vị trí bắt đầu xét, mặc định là 1.
LOCATE(substr, string [, start_position])	
INSTR(substr, string)	

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm tìm kiếm phần tử:

TÊN HÀM	VÍ DỤ
POSITION(substr IN string)	SELECT POSITION (“elo” IN 'Hello World') AS result;
LOCATE(substr, string [, start_position])	SELECT LOCATE (“o”, 'Hello World’, 6) AS result;
INSTR(substr, string)	SELECT LOCATE (“lo”, 'Hello World') AS result;

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm trích xuất và tách chuỗi:

TÊN HÀM	CHỨC NĂNG
SUBSTR(string, start-pos, length)	Trích xuất chuỗi con của **string**, bắt đầu ở vị trí **start-pos** (có thể ở giữa chuỗi) với độ dài chuỗi con là **length**.
LEFT(string, n)	Lấy **n** ký tự từ phần đầu trái/phải của chuỗi → hữu ích khi tất cả các giá trị **string** đều tuân theo một mẫu nhất định và ta cần lấy một phần của chuỗi dựa trên các mẫu đó.
RIGHT(string, n)	

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm tìm kiếm phần tử:

TÊN HÀM	VÍ DỤ
SUBSTR(string, start-pos, length)	SELECT SUBSTR ('Hello World', 2, 4) AS result;
LEFT(string, n)	SELECT LEFT ('Hello World', 2) AS result;
RIGHT(string, n)	SELECT RIGHT ('Hello World', 2) AS result;

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm ghép chuỗi:

TÊN HÀM	CHỨC NĂNG
GROUP_CONCAT(Attr1 [ORDER BY Attr2 ASC|DESC])	Kết hợp các chuỗi ở từng bản ghi trong cột **Attr1** và/hoặc sắp xếp chúng theo thứ tự của thuộc tính **Attr2** khi ghép chuỗi.
CONCAT(str_1, …, str_n)	Kết hợp các chuỗi **str_i** lại với nhau thành một chuỗi duy nhất.
Str1 || str2 AS combined_string;	

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

b. VỚI DỮ LIỆU DẠNG CHUỖI – chuẩn hóa chuỗi

- Các hàm tìm kiếm phần tử:

TÊN HÀM	VÍ DỤ
GROUP_CONCAT(Attr1 [ORDER BY Attr2 ASC|DESC])	SELECT GROUP_CONCAT(HoTen ORDER BY Tuoi)
CONCAT(str_1, …, str_n)	SELECT CONCAT (‘World’, 'Hello’) AS result;
Str1 || str2 AS combined_string;	SELECT ‘World’ || 'Hello’ AS result;

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Là một trong những loại dữ liệu phức tạp nhất trong SQL: do sự đa dạng trong định dạng biểu diễn ngày tháng vì có thể được biểu thị ở nhiều định dạng khác nhau:
  - Tháng có thể được biểu thị bằng tên (ví dụ: January) hoặc bằng số.
  - Năm có thể được viết đầy đủ (ví dụ: 2019) hoặc viết tắt (ví dụ: 19).
  - Thứ tự của các thành phần có thể thay đổi (ví dụ: year-month-day, month-day-year, day-month-year).
- Định dạng chuẩn SQL cho:
  - DATE là YYYY-MM-DD. Định dạng này được sắp xếp theo thứ tự thời gian.
  - TIME là HH:MM:SS[.NNNNNNN] (7 chữ số cho phân số của giây - optional).
  - DATETIME (timestamps) là DATE kết hợp với TIME, được phân tách bằng dấu cách: YYYY-MM-DD HH:MM:SS.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Một số hàm khác khởi tạo dữ liệu thời gian trong MYSQL:

TÊN HÀM	Ý NGHĨA
`current_date()`	Ngày hiện tại, dạng date.
`current_time()`	Thời gian hiện tại, dạng time.
`now()`	Ngày và thời gian hiện tại, dạng date và time.
`make_date(year int, month int, day int)`	Tạo một giá trị date.
`maketime(hour int, min int, sec double-precision)`	Tạo một giá trị thời gian.
`EXTRACT('year' FROM cleaned_date)`	Trích xuất các thành phần khác nhau từ một cột dữ liệu (year, month, day, hour, minute, second, decade, dow – thứ trong tuần) → tạo ra cột mới chứa thông tin chi tiết hơn về thời gian

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Định dạng INTERVAL (khoảng thời gian) được chia thành hai loại:
  - YEAR TO MONTH, với định dạng: YYYY-MM.
  - DAY TO FRACTION, với định dạng: DD HH:MM:SS.F.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Cấu trúc của giá trị khoảng thời gian trong POSTGRE được xác định theo mẫu sau:

$$\boldsymbol{quantity\ unit\ [quantity\ unit\ ...]\ [direction]}$$

Trong đó:

- $\boldsymbol{quantity}$: là số hoặc dấu.
- $\boldsymbol{unit}$: đơn vị thời gian (microsecond, millisecond, second, minute, hour, day, week, month, year, decade, century, millennium hoặc các từ viết tắt của chúng).
- $\boldsymbol{direction}$: "ago" hoặc bỏ trống → cho biết khoảng thời gian này là trong tương lai hay quá khứ.
- VD: "1 year 2 months" biểu thị khoảng thời gian 1 năm và 2 tháng; "3 days 4:05:06" biểu thị 3 ngày, 4 giờ, 5 phút và 6 giây.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Cấu trúc của giá trị khoảng thời gian trong MYSQL được xác định theo mẫu sau:

$$INTERVAL\ quantity\ unit$$

Trong đó:

- $INTERVAL$: từ khóa bắt buộc, chỉ khoảng thời gian.
- $quantity$: là số lượng (cho khoảng thời gian).
- $unit$: đơn vị thời gian (microsecond, millisecond, second, minute, hour, day, week, month, year, decade, century, millennium hoặc các từ viết tắt của chúng).

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Phép toán trên dữ liệu thời gian: hầu hết các hệ thống đều hỗ trợ các phép toán trên dữ liệu thời gian, trong đó toán tử "+/-" để cộng/trừ giữa các ngày/khoảng thời gian.
- **VD14**:
  - `SELECT DATE(now()) + INTERVAL 7 day;` → cộng thêm 7 ngày.
  - `SELECT DATE(now()) + INTERVAL 7 hour;` → cộng thêm 1 giờ vào ngày ban đầu, mặc định là nửa đêm.
  - `SELECT DATE(now()) + INTERVAL '03:00' HOUR_MINUTE;` → cộng thêm 3 giờ vào ngày ban đầu, mặc định là nửa đêm.
  - `SELECT DATE(now()) + INTERVAL 1 day + INTERVAL 1 hour;` → cộng thêm 1 ngày và 1 giờ.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Phép toán trên dữ liệu thời gian: hầu hết các hệ thống đều hỗ trợ các phép toán trên dữ liệu thời gian, trong đó toán tử "+/-" để cộng/trừ giữa các ngày/khoảng thời gian.
- Các hàm tính toán dữ liệu thời gian:

TÊN HÀM	Ý NGHĨA
DATE_ADD() hoặc ADDDATE()	cộng một giá trị thời gian vào một ngày.
DATE_DIFF() hoặc DATEDIFF()	trừ một ngày từ một ngày khác. Hàm này trả về sự khác biệt giữa hai ngày theo số ngày.
ADDTIME()	cộng hai giá trị TIME.
DATE_SUB()	trừ một giá trị thời gian từ một ngày (có thể cộng một số âm vào ngày bằng cách sử dụng DATE_ADD()).

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Tùy thuộc vào hệ quản trị CSDL mà có các tên gọi khác nhau để trích xuất từng phần của dữ liệu như:
  - DATE() trích xuất phần ngày,
  - TIME() trích xuất phần giờ,
  - DAYNAME() để xác định ngày trong tuần của một ngày cụ thể, …
- Xử lý NULL bằng cách thay thế bởi giá trị “zero”: '0000-00-00' là giá trị zero dùng để thay thế cho NULL trong trường hợp các giá trị ngày tháng bị thiếu → thay vì sử dụng NULL để biểu thị một ngày không xác định hoặc không có, ta có thể sử dụng '0000-00-00’.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- MYSQL chấp nhận các định dạng khác nhau của ngày tháng:
  - Năm: có thể là 2 hoặc 4 chữ số.
  - DATETIME/ TIMESTAMP ở dạng chuỗi: có thể sử dụng bất kỳ ký tự dấu làm phân cách (ví dụ: '98/09/04', '98@09@04') hoặc có thể không cần dấu phân cách ('980904').
  - DATETIME ở dạng số: có thể biểu diễn giá trị datetime dưới dạng một số, miễn là hệ thống có thể hiểu.

ƯU	NHƯỢC
Dễ dàng nhập dữ liệu từ các nguồn khác nhau với các định dạng ngày tháng khác nhau vào MySQL.	không nhất quán và khó khăn trong việc kiểm soát chất lượng dữ liệu.

⇒ Cần xem xét chuẩn hóa định dạng ngày tháng khi nhập dữ liệu, sử dụng các hàm chuyển đổi để chuyển đổi các chuỗi ngày tháng không chuẩn thành định dạng chuẩn.

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Định dạng giá trị ngày theo chuỗi định dạng **DATE_FORMAT(date, format)**
  - **date** Chuỗi thời gian cần định dạng.
  - **format** Chuỗi kiểu định dạng: sử dụng các ký tự đặc tả bắt đầu bằng dấu phần trăm (%) để chỉ định cách bố trí ngày tháng, VD: %M hiển thị tháng ở dạng tên đầy đủ, %d hiển thị ngày ở dạng số và %Y hiển thị năm ở dạng số có 4 chữ số,…

VD15: `SELECT DATE_FORMAT("2019-08-12", "%M %d %Y");` → `August 12 2019.`

3.2 LÀM SẠCH DỮ LIỆU

3.2.1 BIẾN ĐỔI DỮ LIỆU

c. VỚI DỮ LIỆU DẠNG THỜI GIAN

- Chuyển đổi số nguyên sang giá trị thời gian sử dụng **MAKETIME(h, m, s)**
  - **h** Số giờ.
  - **m** Số phút.
  - **s** Số giây.

VD16: để chuyển đổi giá trị 547 thành thời gian ‘5:47:00’, ta có:

```
SELECT MAKETIME(547 DIV 100, 547 MOD 100, 0);
```

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Xác định các giá trị thiếu** khá phức tạp vì các tập dữ liệu khác nhau sử dụng các phương pháp khác nhau để biểu thị sự thiếu sót dữ liệu:
  - **Không được xác định rõ ràng**: Giá trị bị thiếu đơn giản là không có trong vài trường hợp. Ví dụ: trong tệp CSV, điều này có thể được biểu thị bằng hai dấu phẩy liền kề (,,) hoặc một dòng kết thúc bằng dấu phẩy.
  - **Sử dụng dấu hiệu rõ ràng**: Các tập dữ liệu khác có thể sử dụng các dấu hiệu cụ thể để biểu thị giá trị bị thiếu. Tuy nhiên, không có dấu hiệu "tiêu chuẩn" hoặc "điển hình“ → cần kiểm tra cẩn thận tập dữ liệu.
  - **Các phương pháp phổ biến**: sử dụng các giá trị nằm ngoài phạm vi giá trị (với dữ liệu số) hoặc sử dụng các chuỗi cụ thể. Ví dụ: -1 có thể được sử dụng trong một trường số chỉ chứa các giá trị dương hoặc chuỗi trống (""). Các giá trị phổ biến khác trong các trường phân loại là 'n/a', 'N/A' hoặc 'NA'.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Xử lý các giá trị thiếu** bằng cách thay thế các các giá đánh dấu sự thiếu sót bằng NULL:
  - Đảm bảo tính nhất quán: trong SQL, NULL là cách tiêu chuẩn để biểu thị sự vắng mặt của một giá trị. Bằng cách chuyển đổi các giá trị khác (ví dụ: "-1", "N/A", "") thành NULL → đảm bảo tất cả các giá trị bị thiếu được xử lý thống nhất.
  - Xử lý chính xác dữ liệu thiếu: Các hàm và toán tử SQL thường có ảnh hưởng đặc biệt khi gặp NULL (ví dụ: bỏ qua NULL trong các phép tính tổng hợp). Việc sử dụng NULL một cách nhất quán đảm bảo rằng các phép tính này hoạt động như mong đợi.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Một số vấn đề với NULL:**
  - Kiểm tra các giá trị thay thế khác: cần xác định được tất cả các giá trị đại diện cho dự liệu bị thiếu rồi mới thay thế.
  - Ảnh hưởng của NULL trong mệnh đề WHERE:
    - So sánh với NULL: NULL luôn trả về FALSE / UNKNOWN.
    - Mọi phép tính liên quan đến NULL đều trả về NULL.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Một số vấn đề với NULL:**

VD17: Cho bảng Patient(id, height, weight), hãy xác định những bệnh nhân có chỉ số BMI dưới 18.5, biết:

$$BMI = \frac{weight\ (kg)}{\left(height\ (m)\right)^2}$$

ID	Height(cm)	Weight(kg)
1	170.5	65
2	165.2	70.3
3	180.1	75.5
4	160	55.2
5	170.4	NULL

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Một số vấn đề với NULL:**

VD17: Cho bảng Patient(id, height, weight), hãy xác định những bệnh nhân có chỉ số BMI trên 18.5, biết:

$$BMI = \frac{weight\ (kg)}{\left(height\ (m)\right)^2}$$

```
SELECT *
FROM Patient
WHERE Weight/power(Height/100, 2) > 18.5;
```

ID	Height(cm)	Weight(kg)
1	170.5	65
2	165.2	70.3
3	180.1	75.5
4	160	55.2

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Một số vấn đề với NULL:**
  - Ảnh hưởng của NULL trong mệnh đề kết hợp:

MỆNH ĐỀ	ẢNH HƯỞNG	VÍ DỤ
AND	Nếu một điều kiện trong AND là NULL thì toàn bộ mệnh đề là FALSE.	`SELECT * FROM Patient WHERE Weight > 22 and Height < 200;`
OR	Nếu một trong hai điều kiện là TRUE thì mệnh đề là TRUE; nếu cả hai điều kiện là NULL thì mệnh đề là FALSE.	`SELECT * FROM Patient WHERE Weight > 22 or Height < 200;`
NOT	Không thể kiểm tra phủ định của NULL.	`SELECT * FROM Patient WHERE NOT (Weight <= 22);`

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Một số vấn đề với NULL:**
  - Ảnh hưởng của NULL trong mệnh đề kết hợp:

ID	Height(cm)	Weight(kg)
1	170.5	65
2	165.2	70.3
3	180.1	75.5
4	160	55.2

```
SELECT * FROM Patient
WHERE Weight > 22
      and Height < 200;
```

ID	Height(cm)	Weight(kg)
1	170.5	65
2	165.2	70.3
3	180.1	75.5
4	160	55.2
5	170.4	NULL

```
SELECT * FROM Patient
WHERE Weight > 22 or Height < 200;
```

ID	Height(cm)	Weight(kg)
1	170.5	65
2	165.2	70.3
3	180.1	75.5
4	160	55.2

```
SELECT * FROM Patient
WHERE NOT (Weight <= 22);
```

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Một số vấn đề với NULL:**
  - Ảnh hưởng của NULL trong **GROUP BY**: NULL được coi là một nhóm và được GROUP như bình thường, tuy nhiên tuy thuộc vào cài đặt của hệ quản trị mà sẽ trả về warning.
  - Ảnh hưởng của NULL trong Aggregate Function:

TÊN HÀM	Ý NGHĨA
SUM, AVG, MAX và MIN	Bỏ qua các giá trị NULL.
COUNT(*)	Đếm tất cả các hàng, kể cả NULL.
COUNT(col_name)	Đếm tất cả các hàng trong col_name, loại trừ NULL.

⇨ Để kiểm tra sự tồn tại của NULL, sử dụng các toán tử IS NULL và IS NOT NULL.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Giải pháp xử lý NULL:**
  - Loại bỏ NULL: xóa các hàng/ cột chứa NULL bằng cách sử dụng DELETE FROM … WHERE Attr IS NULL hoặc UPDATE … DROP COLUMN Attr → Nhược: xóa cột có thể gây mất dữ liệu đáng kể.
  - Thay thế NULL: thay bởi một giá trị khác (mean/mode/median/…) hoặc dự đoán giá trị → phức tạp, tùy thuộc vào loại DL; có thể sử dụng COALESCE (attribute, newvalue) để thay thế NULL bằng một giá trị mặc định.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Có 3 kiểu thiếu dữ liệu:** Giả sử một bảng có thuộc tính A có chứa một vài giá trị thiếu và B là các thuộc tính khác của bảng.
  - Kiểu 1 - Các giá trị bị thiếu hoàn toàn ngẫu nhiên: giá trị bị thiếu hoàn toàn ngẫu nhiên, không liên quan gì đến các giá trị của thuộc tính A hay mọi thuộc tính khác trong bộ dữ liệu, do đó các giá trị hiện có của A là một biểu diễn không thiên vị, cho ý tưởng tốt về sự phân phối của tất cả các giá trị trong A ➔ giải quyết: điền vào các giá trị thiếu bởi GTTB hoặc khớp một phân phối để xem giá trị nào của A có khả năng bị thiếu.

VD18.1: Nếu cảm biến đo nhiệt độ không hoạt động do hết pin mà không liên quan đến thời tiết, có thể thay thế giá trị thiếu bằng trung bình nhiệt độ, vì đây là một ước lượng không chệch.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Có 3 kiểu thiếu dữ liệu:** Giả sử một bảng có thuộc tính A có chứa một vài giá trị thiếu và B là các thuộc tính khác của bảng.
  - Kiểu 2 - Các giá trị bị thiếu ngẫu nhiên: giá trị bị thiếu không phụ thuộc vào các giá trị của A nhưng có thể phụ thuộc vào các giá trị khác (B) → có yếu tố liên quan gián tiếp. Nghĩa là A có tương quan với một/nhiều thuộc tính khác → giải quyết: điền vào các giá trị thiếu nếu tìm ra được thuộc tính nào đó có liên quan đến A và suy ra mối quan hệ giữa chúng.

VD18.2: Nếu pin của cảm biến không được thay thì khi trời mưa, giá trị nhiệt độ bị thiếu sẽ liên quan đến thời tiết mưa mà không liên quan trực tiếp đến nhiệt độ → có thể sử dụng thông tin từ thuộc tính đó (ví dụ: lấy trung bình nhiệt độ ngày mưa hoặc dùng hồi quy tuyến tính) để suy luận giá trị thiếu.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Có 3 kiểu thiếu dữ liệu:** Giả sử một bảng có thuộc tính A có chứa một vài giá trị thiếu và B là các thuộc tính khác của bảng.
  - ➢ Kiểu 3 - Các giá trị bị thiếu không thể bỏ qua: là các giá trị độc lập với các giá trị khác (B) nhưng có thể phụ thuộc vào giá trị cơ bản của A → không thể điền giá trị bị thiếu vì các giá trị của thuộc tính khác không giúp ích trong việc tạo giá trị mới cũng như nó có mối liên hệ với các giá trị khác trong A.

VD18.3: Nếu cảm biến hỏng khi nhiệt độ dưới 0°C, thì giá trị bị thiếu chủ yếu thuộc về nhiệt độ thấp. Trong trường hợp này, không thể đơn giản nội suy từ các giá trị khác vì dữ liệu bị thiếu chỉ nằm trong một phần nhỏ của khoảng nhiệt độ, khiến việc ước lượng chính xác trở nên khó khăn.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

a. THIẾU DỮ LIỆU

- **Nhận biết các loại giá trị bị thiếu:** xác định mối quan hệ giữa thuộc tính bị thiếu giá trị với các thuộc tính khác trong bảng → sử dụng tương quan (mặc dù chỉ phát hiện các mối quan hệ tuyến tính) hoặc PMI. Nếu cả hai cách này đều chưa trả lời được câu hỏi liệu B có liên quan lên A không, ví dụ B là dạng số, khi đó ta có thể thử nghiệm sơ bộ là so sánh các giá trị của B khi có A với các giá trị của B khi không có A như sau:

```
SELECT avg(B) as mean,
       CASE (WHEN A IS NULL THEN ‘absent’
             ELSE ‘present’ END) AS Avalue
FROM Data
GROUP BY Avalue;
```

Nếu GTTB có sự khác biệt đáng kể

→ có sự kết nối giữa chúng.

Ngược lại → có thể loại trừ trường hợp 1 và 3; rất khó để phân biệt được hai trường hợp này nếu chỉ dựa vào dữ liệu.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

- Giá trị ngoại lai là một giá trị bất thường xuất hiện trong một thuộc tính của tập dữ liệu (không có khái niệm chính thức về outlier vì còn phụ thuộc vào ngữ cảnh).
  - **Với dữ liệu dạng số**: là các giá trị rất lớn/ rất nhỏ so với phần lớn các giá trị khác trong tập dữ liệu. Ví dụ, nếu nhiệt độ trung bình là 75 độ F và độ lệch chuẩn là 5 độ, thì 100 có thể được coi là một giá trị ngoại lệ vì nó khác xa so với các giá trị thông thường. Để xác định giá trị ngoại lệ cho kiểu dữ liệu số: tìm phân phối phù hợp với dữ liệu và sử dụng độ lệch chuẩn → xác định các giá trị nằm ngoài phạm vi bình thường.
  - **Với dữ liệu dạng phân loại**: là những giá trị xuất hiện rất ít so với các giá trị khác → tính tần suất cho mỗi giá trị và giá trị ngoại lệ sẽ là giá trị có tần suất rất thấp. Ví dụ, ta có một thuộc tính phân loại về màu sắc của sản phẩm, hầu hết các sản phẩm đều có màu xanh, đỏ hoặc vàng. Một sản phẩm màu tím có thể được coi là một giá trị ngoại lệ.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

- Ý nghĩa:
  - Chỉ ra các vấn đề về chất lượng dữ liệu:
    - Lỗi trong quá trình thu thập/biểu diễn dữ liệu.
    - Các tình huống bất thường: sự biến động ngẫu nhiên, một tình huống thực sự không thường xuyên, đặc biệt hoặc bất thường,…
    - Việc xử lý chúng có thể ảnh hưởng đến kết quả phân tích dữ liệu.
  - Trong một số trường hợp, giá trị ngoại lai là những gì ta tìm kiếm. VD: trong gian lận thẻ tín dụng, các giao dịch gian lận là giá trị ngoại lai trong số lượng lớn các giao dịch hợp pháp.
  - Phần lớn trong thực tế, giá trị ngoại lai là các giá trị xấu làm xáo trộn bản chất chung của dữ liệu và cần loại bỏ chúng.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

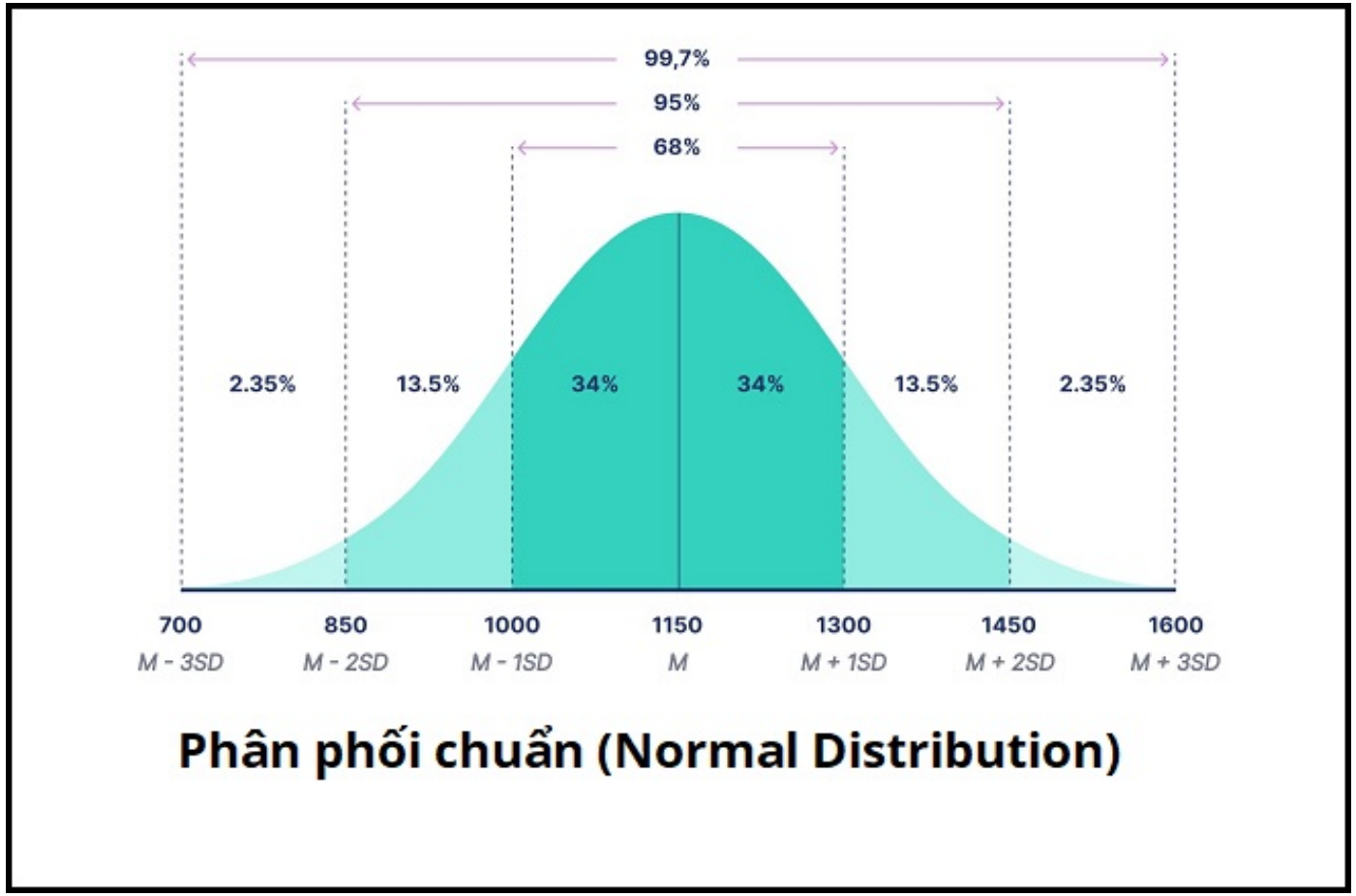
- Phát hiện giá trị ngoại lai cho các thuộc tính đơn lẻ phụ thuộc vào kiểu dữ liệu:
  - Thuộc tính định danh – nominal: là dữ liệu không có thứ tự → tính tần suất cho mỗi loại; giá trị ngoại lai sẽ là một giá trị có tần suất rất thấp.
  - Thuộc tính dạng số: cần giả định một phân phối cơ bản, phù hợp với dữ liệu để xác định được outlier. Khi một phân phối phù hợp với dữ liệu ngoại trừ một vài giá trị → những giá trị đó có thể được coi là giá trị ngoại lệ ➔ thường sử dụng phân phối chuẩn nhưng không phải lúc nào cũng phù hợp, trong đó:
    - 95% giá trị nằm trong khoảng 2 độ lệch chuẩn so với giá trị trung bình.
    - 99% giá trị nằm trong khoảng 3 độ lệch chuẩn.
    - Giá trị nằm ngoài khoảng này có thể được coi là ngoại lai.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

→ Không thể phân biệt ngoại lai với các phân phối khác như phân phối chuẩn.



Phân phối chuẩn (Normal Distribution)

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

- Ảnh hưởng của ngoại lai: ảnh hưởng đến các thống kê mà ta sử dụng: thay đổi giá trị trung bình và làm cho độ lệch chuẩn lớn hơn nhiều → nên sử dụng các phương pháp thống kê mạnh mẽ hơn như **trimmed mean** hoặc **độ lệch tuyệt đối trung vị** (Median Absolute Deviation - MAD) là trung vị của giá trị tuyệt đối của sự sại khác giữa mỗi giá trị và trung vị.
- Trong một số trường hợp, có thể phải thực hiện phân tích đầy đủ như phân cụm (clustering) để quyết định xem đó có là outlier hay không.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

- **VD19**: Cho bảng *people(family_name,...),* ta nghi ngờ có một số bản ghi bị nhập sai (lỗi chính tả, phát âm sai). Một cách để kiểm tra là tập trung vào các giá trị rất hiếm xuất hiện (vì lỗi có xu hướng khác nhau, riêng lỗi sai chính tả đã cho ra rất nhiều kết quả khác nhau). Câu truy vấn sau sẽ hiển thị các tên chỉ xuất hiện một lần:

```
SELECT family_name, count(*) as freq
FROM Dataset
GROUP BY family_name
HAVING freq = 1;
```

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

- **VD20**: tìm outlier trong dữ liệu số: thay vì chỉ tìm các số nằm ngoài phạm vi n độ lệch chuẩn so với GTTB thì để tìm outlier thì ta có thể sử dụng trimmed mean và trimmed standard deviation → giúp xác định outliers đáng tin cậy hơn vì chúng ít bị ảnh hưởng bởi các giá trị cực trị.

3.2 LÀM SẠCH DỮ LIỆU

3.2.2 XỬ LÝ DỮ LIỆU THIẾU VÀ CÁC OUTLIER

b. OUTLIER - GIÁ TRỊ NGOẠI LAI

```
WITH TrimmedStats AS
    (SELECT avg(A) as tmean, stdev(A) as tdev
     FROM Data, (SELECT max(A) as Amax FROM Data) as T1, (SELECT min(A) as Amin FROM
    Data)
     WHERE A < Amax and A > Amin)
SELECT A
FROM Data
WHERE A < tmean - (2*tdev) or A > tmean + (2*tdev);
```

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Xác định các bản ghi trùng lặp trong dữ liệu dạng bảng là nhiệm vụ kiểm tra xem trong dữ liệu có nhiều hơn một hàng tham chiếu đến cùng một quan sát, đối tượng hay thực thể hay không.
  - Xác định một tập các thuộc tính đóng vai trò là khóa chính, nếu hai hay nhiều hàng có các giá trị giống hệt nhau trên các thuộc tính này → trùng lặp dữ liệu.
  - Nếu khóa chính được tạo một cách "nhân tạo" như tự động tăng ID → không thể áp dụng cách trên. Các thuộc tính tự nhiên (VD tên, địa chỉ, ngày sinh,…) có thể áp dụng cách trên.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- **Nhận xét**: xác định các bản ghi trùng lặp không phải lúc nào cũng đơn giản, trong nhiều trường hợp, các bản ghi trùng lặp có thể không có giá trị hoàn toàn giống nhau cho tất cả các thuộc tính.
- Ví dụ: có thể có lỗi chính tả, lỗi đo lường hoặc dữ liệu không chính xác khác trong cơ sở dữ liệu → cần sử dụng các kỹ thuật phức tạp hơn như **fuzzy matching** (khớp gần đúng) để xác định các bản ghi có khả năng trùng lặp không, nghĩa là nếu hai giá trị rất gần nhau .

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- **Khái niệm "gần":** thường được thực hiện bằng cách sử dụng khoảng cách:
  - Khoảng cách nhỏ → các giá trị là tương tự nhau.
  - Ngược lại → các giá trị là khác nhau.

Dữ liệu dạng số	Dữ liệu dạng phân loại
Khoảng cách được đo bằng trị tuyệt đối mức độ sai khác (đôi khi được chuẩn hóa).	Fuzzy matching sẽ coi hai chuỗi là tương tự nhau nếu chúng có nhiều điểm tương đồng hơn so với điểm khác nhau → có thể sử dụng **LIKE** để so sánh định dạng

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Khi so sánh hai bản ghi:
  - Xác định những thuộc tính nào có thể chỉ ra tính duy nhất của từng bản ghi.
  - Với từng bản ghi, chọn một vài công thức tính khoảng cách.
  - Kết hợp các loại khoảng cách đó với nhau để quyết định hai giá trị khi nào là “đủ gần”.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Giả sử $A_1, A_2, \ldots, A_n$ là các thuộc tính được so sánh với nhau, $dist_i$ là khoảng cách của A_i. Hai bản ghi r và s được so sánh bằng cách sử dụng $dist_i$ trên $r.A_1$ và $s.A_1$, … tương tự, $dist_n$ trên $r.A_n$ và $s.A_n$. Tính khoảng cách tổng thể giữa s và r, $dist(s, r)$ bằng các cách sau:
  - Trực tiếp: sử dụng công thức khoảng cách chuẩn hóa tổng thể.
  - Gián tiếp - kết hợp Boolean: tính khoảng cách kết hợp so sánh với ngưỡng.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Tính khoảng cách tổng thể giữa s và r, $dist(s,r)$:
  - **Trực tiếp**: tính khoảng cách giữa các thuộc tính với công thức khoảng cách chuẩn hóa tổng thể

$$dist(r,s) = \frac{\sum_{k=1}^{n} dist_k(r.A_k, s.A_k)}{\sum_{k=1}^{n} \max(dist_k)}$$

Trong đó:

$dist(r,s)$: Khoảng cách tổng thể giữa bản ghi r và s;

$dist_k(r.A_k, s.A_k)$: khoảng cách giữa các giá trị của thuộc tính A_k trong bản ghi r và s;

$max(dist_k)$: khoảng cách tối đa của thuộc tính A_k → được sử dụng để chuẩn hóa khoảng cách cho thuộc tính đó → chuẩn hóa kết quả, đảm bảo khoảng cách tổng thể nằm trong phạm vi nhất định (thường từ 0 đến 1), giúp dễ dàng so sánh khoảng cách giữa các cặp bản ghi khác nhau.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Tính khoảng cách tổng thể giữa s và r, $dist(s,r)$:
  - **Trực tiếp**: tính khoảng cách giữa các thuộc tính với công thức khoảng cách chuẩn hóa tổng thể

$$dist(r,s) = \frac{\sum_{k=1}^{n} dist_k(r.A_k, s.A_k)}{\sum_{k=1}^{n} \max(dist_k)}$$

Tuy nhiên, chỉ phù hợp khi các khoảng cách có cùng thang đo (VD cùng kiểu dữ liệu dạng số đã được chuẩn hóa) → ngược lại, sử dụng chuẩn hóa riêng lẻ:

$$dist(r,s) = \sum_{k=1}^{n} \frac{dist_k(r.A_k, s.A_k)}{\max(dist_k)}$$

➔Khoảng cách tổng thể so sánh với một ngưỡng, nếu $dist(r,s) < threshold$

➔Hai bản ghi giống nhau.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Tính khoảng cách tổng thể giữa s và r, $dist(s, r)$:
  - **Gián tiếp - kết hợp Boolean**: giả sử ta có ngưỡng cho mỗi phép đo khoảng cách, với từng thuộc tính, so sánh các giá trị của chúng với các ngưỡng để xem liệu hai giá trị có tương đồng đủ lớn hay không. Gọi δ_k là mức độ "đủ tốt" giữa hai (hay nhiều) giá trị của một thuộc tính trong các bản ghi, khi đó:

$$\delta_k = \begin{cases} 0, \text{nếu sự tương đồng giữa các giá trị } A_k \text{ "đủ tốt" } (dist < \text{ngưỡng}) \\ 1, \text{nếu sự tương đồng giữa cá c giá trị } A_k \text{ không "đủ tốt" } (dist > \text{ngưỡng}) \end{cases}$$

với các giá trị trong A_k là hai giá trị của thuộc tính đó trong hai bản ghi. Khi đó:

$$dist(r, s) = \frac{\sum_{k=1}^{n} \delta_k(r.A_k, s.A_k)}{n}$$

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Tính khoảng cách tổng thể giữa s và r, $dist(s,r)$:
  - **Gián tiếp - kết hợp Boolean**:

$$dist(r,s) = \frac{\sum_{k=1}^{n} \delta_k(r.A_k, s.A_k)}{n}$$

là tỷ lệ của các thuộc tính “nhất trí” hai bản ghi tương tự nhau khi so sánh với ngưỡng.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Tính khoảng cách tổng thể giữa s và r, $dist(s,r)$:
  - **Gián tiếp - kết hợp Boolean**:

VD21: có hai bản ghi khách hàng và ta muốn xác định xem chúng có phải là cùng một người hay không → so sánh tên, địa chỉ email và số điện thoại của họ.

- Tên: sử dụng khoảng cách Levenshtein, đặt ngưỡng là 2. Nếu khoảng cách Levenshtein giữa hai tên < 2 → tên tương tự nhau (δ_1 = 1).
- Địa chỉ email: sử dụng so sánh chính xác. Nếu hai địa chỉ giống nhau → email tương tự nhau (δ_2 = 1).
- Số điện thoại: sử dụng khoảng cách Hamming, đặt ngưỡng là 1. Nếu khoảng cách Hamming giữa hai số điện thoại nhỏ < 1 → số điện thoại tương tự nhau (δ_3 = 1).
- Tỷ lệ các thuộc tính "nhất trí" hai bản ghi là trùng lặp: dist(r, s) = (δ1 + δ2 + δ3) / 3. Nếu tỷ lệ này vượt quá một ngưỡng nhất định (ví dụ: 0,8), kết luận rằng hai bản ghi có thể là của cùng một người.

3.2 LÀM SẠCH DỮ LIỆU

3.2.3 PHÁT HIỆN VÀ LOẠI BỎ CÁC GIÁ TRỊ TRÙNG LẶP

- Trường hợp đặc biệt: hai bản ghi tham chiếu cùng một đối tượng nhưng lại chứa thông tin mẫu thuẫn nhau.
- Giải quyết:
  - Không thể xác được giá trị nào là đúng → khó khăn → cần có kiến thức về lĩnh vực/ chuyên môn để xử lý sự không nhất quán.
  - Một cách để tránh việc không nhất quán là đặt càng nhiều quy tắc liên quan đến lĩnh vực càng tốt.

3.3 TIỀN XỬ LÝ DỮ LIỆU

- Dữ liệu sau làm sạch vẫn có thể chưa sẵn sàng để phân tích do các giá trị không ở định dạng mà công cụ phân tích/ thuật toán yêu cầu → *Tiền xử lý dữ liệu*: đảm bảo dữ liệu ở định dạng phù hợp để phân tích:

  - **Tổng hợp**: kết hợp hai/ nhiều bản ghi thành một, phổ biến khi dữ liệu có thể xét trên nhiều mức độ cụ thể khác nhau → dữ liệu tổng hợp có ít chi tiết hơn nhưng ít biến động hơn, thấy được tổng thể rõ ràng hơn → sử dụng GROUP BY trong SQL. VD dữ liệu về một sự kiện được thực hiện trong khoảng thời gian đều đặn 1 phút thì có thể xem xét theo khía cạnh giờ bằng cách tổng hợp tất cả các bản ghi trong vòng 60 phút.

  - **Lấy mẫu**: chọn một tập con của dữ liệu → tính toán ít hơn, phổ biến khi muốn thực hiện phân tích phức tạp trên các tập lớn hoặc thử một vài phân tích thăm dò trên cùng một dữ liệu. Chọn mẫu ngẫu nhiên bằng cách lấy một vài hàng dựa trên trình tạo số giả ngẫu nhiên (pseudo-random number) → sử dụng RAND() và LIMIT. VD lấy 10 mẫu ngẫu nhiên SELECT * FROM Dataset ORDER BY RAND() LIMIT 10;

3.3 TIỀN XỬ LÝ DỮ LIỆU

- Sau khi làm sạch dữ liệu thì dữ liệu vẫn có thể chưa sẵn sàng để đưa vào phân tích do các giá trị không ở định ở định dạng mà công cụ phân tích/ thuật toán yêu cầu → tiền xử lý dữ liệu: đảm bảo dữ liệu ở định dạng phù hợp để phân tích:

  - **Giảm chiều:** loại bỏ các thuộc tính không liên quan/ dư thừa khỏi tập dữ liệu (PCA,…) → cải thiện hiệu suất của các thuật toán phân tích, giảm nguy cơ overfitting.
  - **Tạo đặc trưng**: tạo các thuộc tính mới dựa trên thuộc tính đã có → cải thiện độ chính xác của thuật toán phân tích bằng cách cung cấp thông tin mới hữu ích hơn.
  - **Rời rạc hóa**: chuyển đổi các thuộc tính kiểu số liên tục thành các thuộc tính phân loại rời rạc.
  - **Nhị phân hóa**: chuyển đổi một thuộc tính số liên tục (hoặc phân loại) thành một hoặc nhiều thuộc tính nhị phân (0 hoặc 1) → là quá trình tạo biến giả - dummy variable - trong ngữ cảnh thống kê.

3.3 TIỀN XỬ LÝ DỮ LIỆU

VD22: bảng Income(PersonID, yearly-income) là một bảng lớn và thuộc tính yearly-income có phần đuôi (phân phối lệch) chứa các giá trị rất nhỏ. Hãy tạo một bảng NewIncome có cùng thuộc tính và dữ liệu với Income nhưng không chứa các bản ghi nằm ở phần đuôi của phân phối (chọn một điểm cắt/ ngưỡng để xác định điểm bắt đầu của phần đuôi) mà thêm một cặp (id, v) mới, trong đó id được khởi tạo mới, v là GTTB của tất cả các giá trị của yearly-income nằm trong phần đuôi.

- Tóm tắt yêu cầu:
  - Giữ lại các bản ghi có thu nhập hàng năm (yearly-income) lớn hơn một ngưỡng nhất định (cutpoint);
  - Thay thế các bản ghi có thu nhập nhỏ hơn ngưỡng này bằng một bản ghi duy nhất chứa ID mới và giá trị thu nhập trung bình của các bản ghi bị loại bỏ.

BÀI TẬP

Câu 1: Hãy viết câu lệnh SQL để tính sự tương quan giữa A và B theo công thức sau:

$$r_{AB} = \frac{n\Sigma(a_i b_i) - (\Sigma a_i \Sigma b_i)}{\sqrt{n\Sigma a_i^2 - (\Sigma a_i)^2}\sqrt{n\Sigma b_i^2 - (\Sigma b_i)^2}}.$$

Câu 2: Một công ty oto đang kiểm tra 3 loại mẫu mới A, B và C trong 4 ngày, và chấm điểm theo thang từ 1 đến 10 điểm cho mỗi ngày với bảng sau. Liệu có sự khác biệt đáng kể giữa các mẫu dựa trên điểm số mà chúng nhận được trong 4 ngày thử nghiệm không? Kết quả thử nghiệm phụ thuộc vào ngày hay phụ thuộc vào mẫu xe? Hãy chuyển đổi dữ liệu sang dạng quan hệ và thực hiện kiểm tra χ2.

	A	B	C
Day 1	8	9	7
Day 2	7.5	8.5	7
Day 3	6	7	8
Day 4	7	6	5

BÀI TẬP

Câu 3: Bảng flights(departure_time,…) chứa các giá trị thời gian dưới dạng số nguyên (ví dụ: 830 cho 8:30 AM, 1445 cho 2:45 PM). Hãy chuyển đổi các giá trị này thành định dạng thời gian.

Câu 4: Viết truy vấn SQL để tìm các ngoại lệ bằng cách sử dụng MAD. Một quy tắc chung là xem xét các giá trị ngoại lệ lớn hơn 1,5 lần so với giá trị MAD, trong đó x là số độ lệch chuẩn mà ta coi là có ý nghĩa.

Câu 5: Hãy xác định liệu hai người trong bảng Patient(last_name, weight, height) có phải là một người hay không bằng cách sử dụng khoảng cách kết hợp Boolean trên “last_name” và “weight”.

TỔNG KẾT

- Làm sạch dữ liệu bao gồm:
  - Biến đổi dữ liệu: chuyển đổi kiểu dữ liệu của các biến (ví dụ từ chuỗi ký tự sang số hoặc từ số sang danh mục) để đảm bảo tính đồng nhất và phù hợp cho quá trình phân tích và mô hình hóa.
  - Xử lý dữ liệu thiếu và giá trị ngoại lai: các phương pháp phổ biến như thay thế giá trị thiếu bằng giá trị trung bình, trung vị, hoặc giá trị ước lượng; loại bỏ hoặc xử lý giá trị ngoại lệ tùy từng trường hợp.
  - Phát hiện và loại bỏ trùng lặp: xác định và loại bỏ các bản ghi trùng lặp trong tập dữ liệu để đảm bảo tính duy nhất và chính xác của dữ liệu.

- Tiền xử lý dữ liệu: chuẩn bị dữ liệu trước khi phân tích và mô hình hóa: chuẩn hóa, tỷ lệ hóa (scaling), mã hóa và trích xuất/ trích chọn đặc trưng (feature extraction) → giúp tối ưu hóa dữ liệu và tăng cường độ chính xác của mô hình.

TỔNG KẾT

- Thăm dò/ khám phá dữ liệu bao gồm các khía cạnh sau:
  - Phân tích đơn biến: phân tích từng biến riêng lẻ nhằm hiểu rõ hơn về từng biến, bao gồm GTTB, độ lệch chuẩn, phân phối và các giá trị ngoại lai, với các công cụ hỗ trợ như biểu đồ histogram, boxplot và biểu đồ phân phối tần suất.
  - Phân tích đơn biến: phân tích mối quan hệ giữa haihay nhiều biến, giúp nhận diện mối quan hệ tương quan hay mức độ phụ thuộc giữa các biến, các công cụ có thể kể đến như biểu đồ scatter plot, heatmap, và phân tích hồi quy.
  - Khớp phân phối: kiểm tra và so khớp dữ liệu với các phân phối lý thuyết khác nhau để tìm ra phân phối phù hợp nhất với dữ liệu.