네, 정확합니다! **LENGTH - LENGTH(REPLACE)** 패턴은 SQLD 시험뿐만 아니라 실무에서도 정규표현식을 쓸 수 없는 환경(구버전 DB 등)에서 문자열 내 특정 패턴의 빈도를 세는 표준 공식입니다.

질문하신 대로 이 로직을 응용하거나, 옛날 방식(Legacy)으로 문자열을 다루는 "고인물들의 알고리즘" 몇 가지를 소개해 드릴게요. 이 원리를 알면 데이터 핸들링 능력이 훨씬 좋아집니다.

---

1. LENGTH 응용 심화: "단어가 몇 번 나왔는지 세기"

지금까지는 문자 하나(a 또는 줄바꿈)만 셌죠? 만약 **'apple'**이라는 단어가 몇 번 나왔는지 세려면 어떻게 해야 할까요?

단순히 빼면 안 됩니다. 'apple'은 5글자니까, 한 번 지워질 때마다 길이가 5씩 줄어들기 때문입니다. 그래서 단어의 길이로 나누어주는 과정이 추가되어야 합니다.

공식:

$$(\text{전체길이} - \text{단어를 지운 길이}) \div \text{단어의 길이}$$

SQL 예시:

SQL

```
SELECT
   (LENGTH(txt) - LENGTH(REPLACE(txt, 'apple', ''))) / LENGTH('apple') AS cnt
FROM table;
```

- apple이 2번 들어있다면?
  - 전체 길이가 10 줄어듦.
  - 10 / 5(apple 길이) = **2**

  SQLD 팁: 시험에서 가끔 함정 문제로 길이가 2 이상인 문자열(예: 'SK', 'LG')의 개수를 물어볼 때가 있습니다. 이때 꼭 나누기를 해줘야 정답입니다.

---

2. 데이터 정합성 검증 (Validation)

이메일이나 전화번호 형식이 맞는지 빠르게 확인할 때도 쓰입니다.

- 이메일에 **'@'**가 정확히 하나만 있는가?
  SQL

```
-- 길이가 딱 1만큼 차이나야 정상
WHERE (LENGTH(email) - LENGTH(REPLACE(email, '@', ''))) = 1
```

- 전화번호**(010-1234-5678)**에 하이픈**(-)**이 **2**개 있는가**?**
  SQL
  ```
  WHERE (LENGTH(phone) - LENGTH(REPLACE(phone, '-', ''))) = 2
  ```

---

3. 레거시 알고리즘**: INSTR + SUBSTR (**위치 찾아서 자르기**)**

SPLIT 함수가 없는 환경에서, **"두 번째 쉼표와 세 번째 쉼표 사이의 글자"**를 가져오고 싶을 때 사용하는 고전적인 방법입니다. 소위 **"발라내기"** 알고리즘입니다.

Python의 find()와 slice를 SQL 수동으로 구현하는 것입니다.

함수 설명**:**

- INSTR(문자열, 찾을단어): 단어가 몇 번째에 있는지 위치(인덱스)를 반환 (Python의 find)
- SUBSTR(문자열, 시작위치, 개수): 문자열을 자름 (Python의 슬라이싱 [:])

알고리즘 논리 **(**예**: 'A,B,C'** 에서 **B**를 꺼내라**):**

1. 첫 번째 쉼표의 위치를 찾는다. (INSTR) -> 2
2. 두 번째 쉼표의 위치를 찾는다. (INSTR 응용) -> 4
3. 첫 번째 쉼표 다음부터**, (**두 위치의 차이 **- 1)**만큼 자른다**.**

SQL 예시 **(Oracle/MySQL):**

SQL

```
SELECT
  SUBSTR(
    txt,
    INSTR(txt, ',') + 1,  -- 첫 쉼표 바로 다음부터
    INSTR(txt, ',', 1, 2) - INSTR(txt, ',') - 1 -- (두 번째 쉼표 위치 - 첫 번째 쉼표 위치 - 1) 길이만큼
  )
FROM table;
```

- 해석**:** 좀 복잡해 보이죠? 하지만 SPLIT이 없는 DB에서는 이 방식으로 텍스트 데이터를 파싱합니다. SQLD 문제에서 INSTR과 SUBSTR이 같이 나오면 **"아, 특정 구간을 잘라내려고 하는구나"**라고 이해하시면 됩니다.

---

4. 시각화 응용: RPAD로 그래프 그리기

LENGTH와 RPAD(오른쪽 채우기)를 이용해서 텍스트로 막대그래프를 그릴 수도 있습니다. 옛날 터미널 환경이나 로그 분석에서 쓰던 방식입니다.

SQL

```
SELECT
  emp_name,
  salary,
  RPAD('■', ROUND(salary/1000), '■') AS chart
FROM employees;
```

결과:

Plaintext

```
SMITH   800  ■■■■■■■■
ALLEN   1600 ■■■■■■■■■■■■■■■■■■■■
WARD    1250 ■■■■■■■■■■■■■
```

- 데이터의 크기(salary)를 LENGTH로 치환해서 시각적으로 보여주는 재미있는 응용입니다.

요약

1. **LENGTH - REPLACE**: 개수 세기의 국룰. (단어 길이에 따라 나누기 주의!)
2. 검증**(Validation)**: 특정 문자가 몇 개 포함되었는지 조건절(WHERE)에서 활용.
3. **INSTR + SUBSTR**: SPLIT 없이 원하는 위치의 문자열을 콕 집어내는 정밀 타격 기술.

SQLD 공부하시면서 이런 패턴들을 "왜 이렇게 짰을까?" 고민해보시면, 나중에 Python으로 데이터 전처리를 할 때도 그 원리가 큰 도움이 될 거예요!