이 쿼리들은 PostgreSQL에서 다양한 문자열, 숫자, 날짜 관련 함수를 활용하여 데이터 조작 및 조회를 수행하는 예제들입니다. 아래에 각 쿼리의 설명을 제공합니다.

기본 함수

1. 문자열 길이 세기

```
SELECT LENGTH('HELLO') AS length_hello
      ,CHAR_LENGTH('HELLO') AS char_length_hello
      ,LENGTH('안녕') AS length_annyung
      ,CHAR_LENGTH('안녕') AS char_length_annyung;
```

- **LENGTH**와 **CHAR_LENGTH** 함수를 사용해 문자열의 바이트 길이와 문자 수를 계산합니다.
- 영어 알파벳의 경우, `LENGTH`와 `CHAR_LENGTH` 결과가 동일하지만, 한글은 `LENGTH`가 더 큽니다.

2. 문자열 붙이기

```
SELECT CONCAT('DREAMS', 'COME', 'TRUE') AS concatenated
      ,CONCAT_WS('-', '2023', '01', '29') AS date_with_dash;
```

- **CONCAT** 함수는 여러 문자열을 이어붙입니다.
- **CONCAT_WS**는 지정한 구분자(`'-'`)로 문자열을 연결합니다.

3. 문자열 자르기

```
SELECT LEFT('SQL 수업', 3) AS left_string
      ,RIGHT('SQL 수업', 4) AS right_string
      ,SUBSTRING('SQL 수업' FROM 2 FOR 5) AS substr_2_5
      ,SUBSTRING('SQL 수업' FROM 2) AS substr_2;
```

- **LEFT**와 **RIGHT** 함수는 문자열의 왼쪽 또는 오른쪽에서 지정한 길이만큼 자릅니다.
- **SUBSTRING** 함수는 지정한 위치에서 시작하여 특정 길이만큼 자릅니다.

4. 문자열 분리하기

```
SELECT SPLIT_PART('서울시 양천구 신정동', ' ', 1) AS part_1
      ,SPLIT_PART('서울시 양천구 신정동', ' ', 3) AS part_3;
```

- **SPLIT_PART** 함수는 구분자로 나눈 문자열 중 지정한 부분을 추출합니다.

5. 문자열 패딩하기

```
SELECT LPAD('SQL', 10, '#') AS lpad_string
      ,RPAD('SQL', 5, '*') AS rpad_string;
```

- **LPAD**와 **RPAD** 함수는 문자열을 지정한 길이만큼 채우며, 앞 또는 뒤에 특정 문자를 추가합니다.

6. 문자열 트림하기

```
SELECT LENGTH(TRIM(LEADING FROM ' SQL ')) AS length_trim_leading
      ,LENGTH(TRIM(TRAILING FROM ' SQL ')) AS length_trim_trailing
      ,LENGTH(TRIM(BOTH FROM ' SQL ')) AS length_trim_both;
```

- **TRIM** 함수는 문자열에서 지정한 방향(앞, 뒤, 양쪽)의 공백이나 특정 문자를 제거합니다.

7. 문자열 트림하기 (특정 문자)

```
SELECT TRIM(BOTH 'abc' FROM 'abcSQLabcabc') AS trim_both
      ,TRIM(LEADING 'abc' FROM 'abcSQLabcabc') AS trim_leading
      ,TRIM(TRAILING 'abc' FROM 'abcSQLabcabc') AS trim_trailing;
```

- **TRIM** 함수는 지정한 문자를 문자열의 앞, 뒤 또는 양쪽에서 제거합니다.

8. 문자열 위치 찾기

```
SELECT POSITION('RUST' IN 'SQL JAVA RUST') AS field_position
      ,POSITION('RUST' IN 'SQL,JAVA,RUST') AS find_in_set
      ,POSITION('인생' IN '너의 인생을 살아라') AS instr_position
      ,POSITION('인생' IN '너의 인생을 살아라') AS locate_position;
```

- **POSITION** 함수는 문자열 내에서 특정 문자열의 시작 위치를 반환합니다.

9. 배열 위치 찾기

```
SELECT (ARRAY['SQL', 'JAVA', 'RUST'])[2] AS elt_result;
```

- PostgreSQL의 **ARRAY** 타입에서 특정 위치의 요소를 가져옵니다.

10. 문자열 반복

```
SELECT REPEAT('*', 5) AS repeated_string;
```

- **REPEAT** 함수는 문자열을 지정한 횟수만큼 반복합니다.

11. 문자열 대체

```
SELECT REPLACE('010.1234.5678', '.', '-') AS replaced_string;
```

- **REPLACE** 함수는 문자열 내의 특정 패턴을 다른 패턴으로 대체합니다.

12. 문자열 역순 정렬

```
SELECT REVERSE('OLLEH') AS reversed_string;
```

- **REVERSE** 함수는 문자열을 거꾸로 뒤집습니다.

13. 숫자 올림, 내림, 반올림, 자르기

```
SELECT CEIL(123.56) AS ceiling_value
      ,FLOOR(123.56) AS floor_value
      ,ROUND(123.56) AS rounded_value
      ,ROUND(123.56, 1) AS rounded_value_1
      ,TRUNC(123.56, 1) AS truncated_value;
```

- **CEIL**은 소수점을 올림합니다.
- **FLOOR**는 소수점을 내림합니다.
- **ROUND**는 반올림합니다.
- **TRUNC**는 지정한 소수점 이하 자리에서 숫자를 자릅니다.

14. 숫자 절대값 부호화

```
SELECT ABS(-120) AS abs_negative
      ,ABS(120) AS abs_positive
```

```
      ,SIGN(-120) AS sign_negative
      ,SIGN(120) AS sign_positive;
```

- **ABS**는 숫자의 절대값을 반환합니다.
- **SIGN**은 숫자의 부호를 반환합니다.

15. 몫과 나머지

```
SELECT 103 % 4 AS remainder_mod
      ,MOD(103, 4) AS remainder_mod_function;
```

- **%** 연산자와 **MOD** 함수는 숫자의 나머지를 반환합니다.

16. 제곱근, 랜덤, 라운드

```
SELECT POWER(2, 3) AS power_result
      ,SQRT(16) AS square_root
      ,RANDOM() AS random_value
      ,ROUND(RANDOM() * 100) AS rounded_random_value;
```

- **POWER**는 제곱을 계산합니다.
- **SQRT**는 제곱근을 계산합니다.
- **RANDOM**은 0과 1 사이의 랜덤 값을 생성합니다.
- **ROUND**는 랜덤 값을 반올림하여 정수로 변환합니다.

17. 현재 날짜와 시간

```
SELECT NOW() AS current_datetime
      ,CURRENT_DATE AS current_date
      ,CURRENT_TIME AS current_time;
```

- **NOW**는 현재 날짜와 시간을 반환합니다.
- **CURRENT_DATE**는 현재 날짜를, **CURRENT_TIME**은 현재 시간을 반환합니다.

18. 날짜에서 특정 요소 추출

```
SELECT NOW() AS current_datetime
      ,EXTRACT(YEAR FROM NOW()) AS year
      ,EXTRACT(QUARTER FROM NOW()) AS quarter
      ,EXTRACT(MONTH FROM NOW()) AS month
```

```
      ,EXTRACT(DAY FROM NOW()) AS day
      ,EXTRACT(HOUR FROM NOW()) AS hour
      ,EXTRACT(MINUTE FROM NOW()) AS minute
      ,EXTRACT(SECOND FROM NOW()) AS second;
```

- **EXTRACT** 함수는 날짜 또는 시간에서 특정 요소(연도, 월, 일 등)를 추출합니다.

19. 날짜 차이 계산

```
SELECT NOW() AS current_datetime
      ,DATE_PART('day', '2025-11-20'::date - NOW()) AS date_diff
      ,DATE_PART('day', NOW() - '2025-11-20'::date) AS reverse_date_diff
      ,DATE_PART('year', '2025-11-20'::date - NOW()) AS years_diff
      ,DATE_PART('month', '2025-11-20'::date - NOW()) AS months_diff
      ,DATE_PART('day', '2025-11-20'::date - NOW()) AS days_diff;
```

- **DATE_PART** 함수는 날짜 간의 차이를 특정 단위(일, 월, 년 등)로 반환합니다.

20. 날짜 계산

```
SELECT NOW() AS current_datetime
      ,NOW() + INTERVAL '20 days' AS date_plus_20_days
      ,NOW() + INTERVAL '20 months' AS date_plus_20_months
      ,NOW() - INTERVAL '20 hours' AS date_minus_20_hours;
```

- **INTERVAL**을 사용해 현재 날짜/시간에서 특정 기간을 더하거나 뺄 수 있습니다.

21. 월말일 계산, 지금까지 지난 일자 계산, 월이름 계산, 요일 출력

```
SELECT NOW() AS current_datetime
      ,DATE_TRUNC('MONTH', NOW()) + INTERVAL '1 MONTH - 1 day' AS
last_day_of_month


 ,EXTRACT(DOY FROM NOW()) AS day_of_year
      ,TO_CHAR(NOW(), 'Month') AS month_name
      ,EXTRACT(DOW FROM NOW()) AS weekday;
```

- **DATE_TRUNC**는 날짜를 지정된 단위로 자릅니다.
- **TO_CHAR**는 날짜를 문자열로 변환하며, 형식을 지정할 수 있습니다.
- **EXTRACT(DOY)**는 올해의 몇 번째 날인지, **EXTRACT(DOW)**는 요일을 숫자로 반환합니다.

22. 형변환

```
SELECT CAST('1' AS INTEGER) AS cast_integer
      ,CAST(2 AS CHAR) AS cast_char
      ,'1'::INTEGER AS convert_integer
      ,2::CHAR AS convert_char;
```

- **CAST**와 `::` 연산자는 데이터 타입을 변환합니다.

23. CASE WHEN

```
SELECT CASE WHEN 12500 * 450 > 5000000 THEN '초과달성' ELSE '미달성' END AS
achievement_status;
```

- **CASE WHEN** 문을 사용해 조건에 따라 다른 결과를 반환합니다.

24. NULLIF, COALESCE

```
SELECT COALESCE(1, 0) AS ifnull_1
      ,COALESCE(NULL, 0) AS ifnull_null
      ,COALESCE(NULLIF(1 / NULLIF(0, 0), NULL), 1) AS
ifnull_division_by_zero;
```

- **COALESCE**는 NULL이 아닌 첫 번째 값을 반환합니다.
- **NULLIF**는 두 값이 같으면 NULL을, 아니면 첫 번째 값을 반환합니다.

25. NULLIF

```
SELECT NULLIF(12 * 10, 120) AS nullif_1
      ,NULLIF(12 * 10, 1200) AS nullif_2;
```

- **NULLIF**는 두 값이 같으면 NULL을 반환하고, 다르면 첫 번째 값을 반환합니다.

26. CASE WHEN 복합 조건

```
SELECT CASE WHEN 12500 * 450 > 5000000 THEN '초과달성'
            WHEN 2500 * 450 > 4000000 THEN '달성'
            ELSE '미달성'
       END AS achievement_status;
```

- **CASE WHEN** 문을 사용해 여러 조건을 평가하고, 해당하는 조건에 맞는 결과를 반환합니다.

문제 풀이

문제 1: 회사이름 변경 및 전화번호 포맷팅

```
SELECT company_name
      ,CONCAT('**', SUBSTRING(company_name FROM 3)) AS cust_company2
      ,phone
      ,REPLACE(SUBSTRING(phone FROM 2), ')', '-') AS phone_number2
FROM customer;
```

- 회사 이름의 첫 두 글자를 `**`로 대체하고, 전화번호의 앞자리 `0`을 제거한 뒤 `)`를 `-`로 바꿔서 출력합니다.

문제 2: 주문 총액, 할인액, 최종 주문 금액

```
SELECT order_id
      ,unit_price * quantity AS order_amount
      ,TRUNC((unit_price * quantity * discount)::integer, -1) AS
discount_amount
      ,(unit_price * quantity) - TRUNC((unit_price * quantity *
discount)::integer, -1) AS final_order_amount
FROM order_details;
```

- 주문의 총액, 할인액, 최종 주문 금액을 계산하여 출력합니다.

문제 3: 직원의 나이 및 고용일 관련 계산

```
SELECT name
      ,birth_date
      ,DATE_PART('year', NOW() - birth_date) AS age
      ,hire_date
      ,DATE_PART('day', NOW() - hire_date) AS days_since_hired
      ,hire_date + INTERVAL '500 days' AS days_after_500
FROM employee;
```

- 직원의 나이, 고용된 이후 지난 일수, 고용일로부터 500일이 지난 날짜를 계산하여 출력합니다.

문제 4: 회원 도시 분류 및 마일리지 등급

```
SELECT contact_name
      ,company_name
      ,city
      ,CASE WHEN city LIKE '%특별시' OR city LIKE '%광역시' THEN '대도시' ELSE
'도시' END AS city_type
      ,mileage
      ,CASE WHEN mileage >= 200 THEN 'VVIP 고객'
            WHEN mileage >= 100 THEN 'VIP 고객'
            ELSE '일반 고객'
       END AS mileage_category
FROM customer;
```

- 회원의 도시를 `대도시`와 `도시`로 분류하고, 마일리지에 따라 고객 등급을 결정합니다.

문제 5: 주문 날짜 관련 정보 출력

```
SELECT order_id
      ,cust_id
      ,order_date
      ,EXTRACT(YEAR FROM order_date) AS order_year
      ,EXTRACT(QUARTER FROM order_date) AS order_quarter
      ,EXTRACT(MONTH FROM order_date) AS order_month
      ,EXTRACT(DAY FROM order_date) AS order_day
      ,EXTRACT(DOW FROM order_date) AS order_weekday
      ,CASE EXTRACT(DOW FROM order_date) WHEN 0 THEN 'Sunday'
                                         WHEN 1 THEN 'Monday'
                                         WHEN 2 THEN 'Tuesday'
                                         WHEN 3 THEN 'Wednesday'
                                         WHEN 4 THEN 'Thursday'
                                         WHEN 5 THEN 'Friday'
                                         ELSE 'Saturday' end as order_week
from orders;
```

- 주문 날짜를 기준으로 연도, 분기, 월, 일, 요일 등을 추출하여 출력합니다. 요일은 숫자와 이름으로 출력됩니다.