CHUNJAE EDUCATION

K-digital Training

데이터베이스 구축

AI센터 개발운영팀 김지훈

소개

강좌구성

데이터베이스 구축	
일자	내용
9/30 (월)	SQL의 이해 및 활용
10/2 (수)	SQL의 응용(1)
10/7 (월)	SQL의 응용(2) 관계형 데이터베이스 구현 관계형 데이터베이스 생성 및 활용
10/8 (화)	비관계형 데이터베이스의 구현(1) 비관계형 데이터베이스의 생성 및 활용(1)
10/10 (목)	응용시스템 요구기능에 적합한 SQL 작성 비관계형 데이터베이스의 구현(2) 비관계형 데이터베이스의 생성 및 활용(2)

소개

오늘강의순서안내

데이터베이스 구축			
교시	시작	종료	내용
1	9:00	9:50	강의 진행 내용 소개, 복습, 컬럼 타입의 종류
2	10:00	10:50	문자열, 수치형 데이터와 함께 사용되는 함수
3	11:00	11:50	문자열, 수치형 데이터 사용 함수 실습 및 확인
4	13:00	13:50	날짜형 데이터에서 사용 가능한 함수
5	14:00	14:50	날짜형 데이터 사용 함수 실습 및 확인
6	15:00	15:50	윈도우 함수 활용 및 실습
7	16:00	16:50	JOIN, UNION
8	17:00	17:50	WITH 활용 및 과제 설명

CONTENTS

04

함수

컬럼 타입의 종류

문자열 데이터

string 으로 표시되는 컬럼

VARCHAR
CHAR, CHARACTER
STRING
TEXT
BINARY
VARBINARY

04

함수

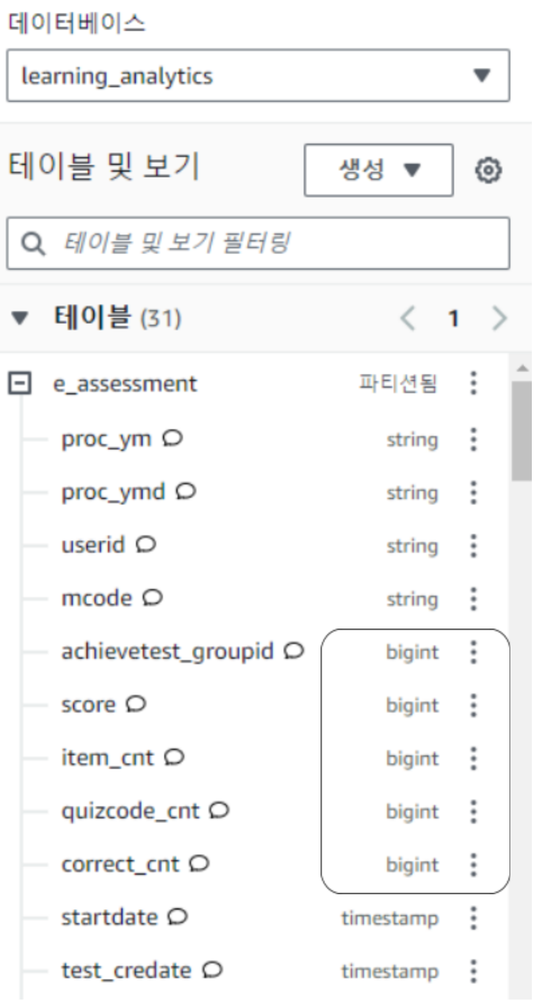
돌아보기

```sql
SELECT
    "userid",
    SUM("point") AS "sum_point"
FROM "learning_analytics"."m_point"
WHERE 1=1
    AND "proc_ym" = '202405'
    AND "point" > 0
GROUP BY "userid"
ORDER BY "sum_point" DESC
LIMIT 100;
```

04

함수

컬럼 타입의 종류



수치형 데이터

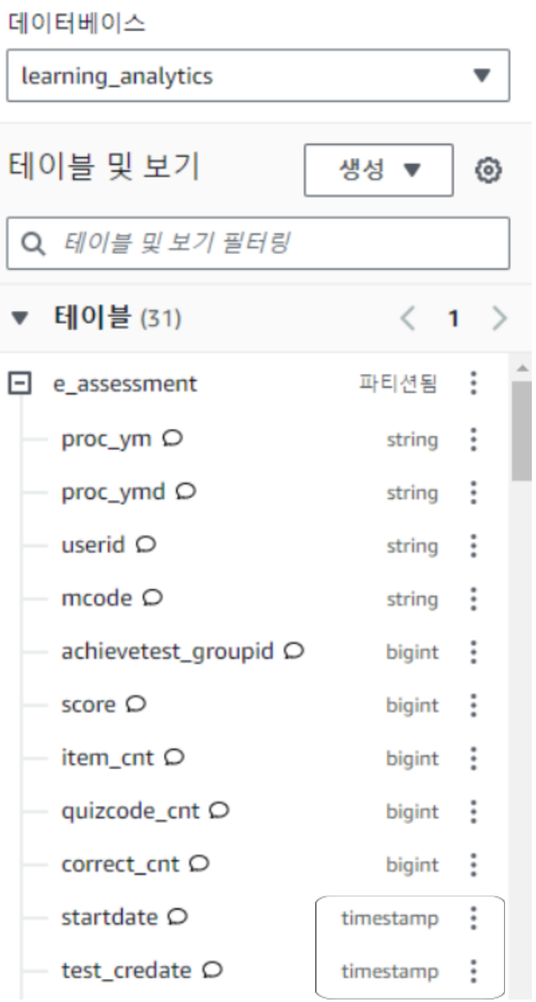
bigint, double 으로 표시되는 컬럼

NUMBER
DECIMAL, NUMERIC
INT, INTEGER, BIGINT, SMALLINT, TINYINT, BYTEINT
FLOAT, FLOAT4, FLOAT8
DOUBLE, DOUBLE PRECISION, REAL

04

함수

컬럼 타입의 종류



날짜형 데이터

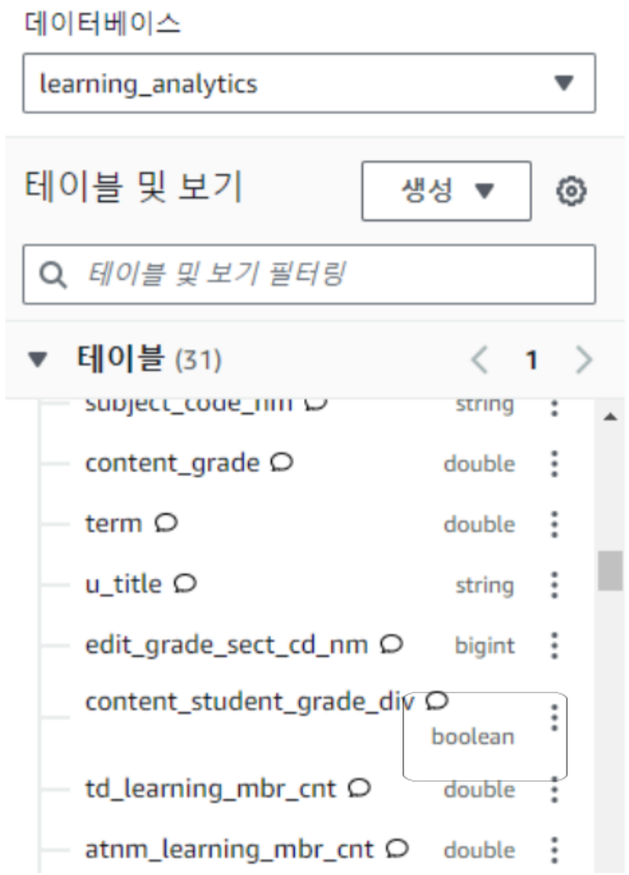
timestamp, date 등으로 표시되는 컬럼

DATE
DATETIME
TIME
TIMESTAMP
TIMESTAMP_LTZ
TIMESTAMP_NTZ
TIMESTAMP_TZ

04

함수

컬럼 타입의 종류



논리형 데이터

boolean 으로 표시되는 컬럼

04

함수

SQL 에서 문자열 데이터와 함께 사용되는 함수

함수	활용 예시	설명
LOCATE	LOCATE("A", "ABC")	"ABC"에서 "A"는 몇 번째에 위치해 있는지 검색해 위치 반환
SUBSTRING	SUBSTRING("ABC", 2)	"ABC"에서 2번째 문자부터 반환
RIGHT	RIGHT("ABC", 1)	"ABC"에서 오른쪽에서 1번째 문자까지 반환
LEFT	LEFT("ABC", 1)	"ABC"에서 왼쪽에서 1번째 문자까지 반환
UPPER	UPPER("abc")	"abc"를 대문자로 바꿔 반환
LOWER	LOWER("ABC")	"ABC"를 소문자로 바꿔 반환
LENGTH	LENGTH("ABC")	"ABC"의 글자 수를 반환
CONCAT	CONCAT("ABC", "DEF")	"ABC" 문자열과 "CDF" 문자열을 합쳐 반환
REPLACE	REPLACE("ABC", "A", "Z")	"ABC"의 "A"를 "Z"로 바꿔 반환

POSITION

LOCATE	LOCATE("A", "ABC")	"ABC"에서 "A"는 몇 번째에 위치해 있는지 검색해 위치 반환

```
SELECT POSITION( 'T' IN 'XTZ' );
SELECT POSITION( 'T' IN 'XYZ' );
SELECT POSITION( 'T' IN 'XTZT');
```

찾고자 하는 글자가 몇 번째에 위치해
있는지 검색하여 위치를 숫자로 반환

SUBSTRING

SUBSTRING	SUBSTRING("ABC", 2)	"ABC"에서 2번째 문자부터 반환
RIGHT	RIGHT("ABC", 1)	"ABC"에서 오른쪽에서 1번째 문자까지 반환
LEFT	LEFT("ABC", 1)	"ABC"에서 왼쪽에서 1번째 문자까지 반환

```
SELECT SUBSTRING('ABC', 2);
SELECT SUBSTRING('CHUNJAE EDUCATION', 9, 3);
SELECT SUBSTRING('CHUNJAE EDUCATION', -3, 3);
```

문자열의 일부를 잘라내어 반환

SUBSTRING(문자열 또는 컬럼, 자르기 시작할 위치, 잘라낼 문자열 길이)

잘라낼 문자열 길이를 지정하지 않으면 문자열 끝까지 전부 반환함

UPPER

```
SELECT UPPER('abc');
```

문자열을 대문자로 바꿔 반환

LOWER

```
SELECT LOWER('ABC');
```

문자열을 소문자로 바꿔 반환

LENGTH

```
SELECT LENGTH('CHUNJAE EDUCATION');
```

문자열의 글자 수를 반환
단, 공백도 글자 하나로 친다.

CONCAT

```
SELECT CONCAT('abcde', '12345', '/*&&');
```

여러 문자열을 하나로 합쳐 반환
숫자도 문자열이기만 하면 합칠 수 있음

REPLACE

```
SELECT REPLACE('TEXTBOOK', 'X', 'S');
SELECT REPLACE('TXTFILES', 'S', '');
```

지정한 문자열을 바꿔 반환

L(R)PAD

```
SELECT LPAD('ABC', 5, 'X'), RPAD('ABC', 5, 'X');
```

```
SELECT LPAD('ABC', 6, 'XYD'), RPAD('ABC', 6, 'XY');
```

지정한 문자열의 길이만큼
특정 문자열을 채워 반환

04

함수

SQL 에서 수치형 데이터와 함께 사용되는 함수

함수	활용	설명
ABS	ABS(숫자)	숫자의 절댓값 반환
CEILING	CEILING(숫자)	숫자를 정수로 올림해서 반환
FLOOR	FLOOR(숫자)	숫자를 정수로 내림해서 반환
ROUND	ROUND(숫자, 자릿수)	숫자를 소수점 자릿수까지 반올림해서 반환
TRUNCATE	TRUNCATE(숫자, 자릿수)	숫자를 소수점 자릿수까지 버림 해서 반환
POWER	POWER(숫자A, 숫자B)	숫자A의 숫자B 제곱 반환
MOD	MOD(숫자A, 숫자B)	숫자A를 숫자B로 나눈 나머지 반환

ABS

```
SELECT ABS(5.24), ABS(-5.43);
```

숫자의 절댓값을 반환

CEILING

```
SELECT CEILING(44.552);
```

숫자를 정수로 올림해 반환

수치형 데이터와 함께 사용되는 함수 - FLOOR

FLOOR

```
SELECT FLOOR(44.552);
SELECT FLOOR(-2.54);
```

숫자를 정수로 내림해 반환

ROUND

```
SELECT ROUND(44.552, 2);
```

숫자를 해당 소수점 자리 수까지
반올림해서 반환

TRUNCATE

```
SELECT TRUNCATE(448.5525);
```

숫자에서 소수점 이하를
전부 버림하여 반환

POWER

```sql
SELECT POWER(5, 2);
```

앞의 숫자를 뒤의 숫자만큼
제곱하여 반환

MOD

```
SELECT MOD(5, 2), 5%2;
```

앞의 숫자를 뒤의 숫자로 나눈
'나머지'를 반환

04

함수

실습 4-1

e_media 테이블 확인 후 함수 사용해보기

e_media 테이블에는
년도를 의미하는 yyyy 컬럼과
월을 의미하는 mm 컬럼이 있습니다.

proc_ymd 컬럼을 사용하여
일자를 확인할 수 있는 dd 컬럼을 만들어 주세요.
(proc_ymd가 20230726 이라면 dd 컬럼 값은 26입니다.)
출력시 proc_ymd, yyyy, mm, dd 컬럼 4개를
순서대로 출력하세요

04

함수

실습 4-1

e_media 테이블 확인 후 함수 사용해보기 I

```
SELECT
    "proc_ymd", "yyyy", "mm",
    SUBSTRING("proc_ymd", 7, 2) AS "dd"
FROM "learning_analytics"."e_media"
LIMIT 5;

SELECT
    "proc_ymd", "yyyy", "mm",
    SUBSTRING("proc_ymd", -2, 2) AS "dd"
FROM "learning_analytics"."e_media"
LIMIT 5;
```

04

함수

실습 4-2

e_media 테이블 확인 후 함수 사용해보기

e_media 테이블에는
전체 액션 수를 의미하는 media_action_cnt 와
비디오 점프 횟수를 의미하는 video_jump_cnt 가 있습니다.
전체 중 비디오 점프 횟수가 차지하는 비율을 확인할 수 있도록
컬럼을 만들고 그 값을 소수점 둘째 자리까지 반올림 해 주세요.
그리고 해당 컬럼 이름을 "jump_rate" 로 해 주세요.
media_action_cnt, video_jump_cnt, jump_rate 순으로
출력해 주세요.

04

함수

실습 4-2

e_media 테이블 확인 후 함수 사용해보기 I

```
SELECT
    "media_action_cnt", "video_jump_cnt",
    ROUND("video_jump_cnt"/CAST("media_action_cnt" AS DOUBLE), 2) AS "jump_rate"
FROM "learning_analytics"."e_media"
WHERE 1=1
    AND "media_action_cnt" IS NOT NULL
LIMIT 50;
```

04

함수

날짜형 데이터의 구성

```
SELECT NOW() AS "현재"
    , YEAR(NOW()) AS "연"
    , QUARTER(NOW()) AS "분기"
    , MONTH(NOW()) AS "월"
    , WEEK(NOW()) AS "주"
    , DAY(NOW()) AS "일"
    , HOUR(NOW()) AS "시"
    , MINUTE(NOW()) AS "분"
    , SECOND(NOW()) AS "초"
    , MILLISECOND(NOW()) AS "밀리초"
;

SELECT NOW()
    , DAY_OF_YEAR(NOW()) AS "연 중 몇번째 일"
    , DAY_OF_WEEK(NOW()) AS "주 중 몇번째 일"
;
```

04

함수

날짜형 데이터의 버림

```
SELECT NOW() AS "현재"
    , DATE_TRUNC('year', NOW())
    , DATE_TRUNC('quarter', NOW())
    , DATE_TRUNC('month', NOW())
    , DATE_TRUNC('week', NOW())
    , DATE_TRUNC('day', NOW())
    , DATE_TRUNC('hour', NOW())
    , DATE_TRUNC('minute', NOW())
    , DATE_TRUNC('second', NOW())
    , DATE_TRUNC('millisecond', NOW())
;
```

04

함수

날짜형 데이터의 연산

```
SELECT NOW() AS "현재"
    , DATE_ADD('year', 1, NOW())
    , DATE_ADD('quarter', 1, NOW())
    , DATE_ADD('month', 1, NOW())
    , DATE_ADD('week', 1, NOW())
    , DATE_ADD('day', 1, NOW())
    , DATE_ADD('hour', 1, NOW())
    , DATE_ADD('minute', 1, NOW())
    , DATE_ADD('second', 1, NOW())
    , DATE_ADD('millisecond', 1, NOW())
;
```

04

함수

두 날짜형 데이터의 간격

```
SELECT NOW() AS "현재"
    , DATE_DIFF('year', DATE_ADD('year', 1, NOW()), NOW())
    , DATE_DIFF('quarter', DATE_ADD('quarter', 1, NOW()), NOW())
    , DATE_DIFF('month', DATE_ADD('month', 1, NOW()), NOW())
    , DATE_DIFF('week', DATE_ADD('week', 1, NOW()), NOW())
    , DATE_DIFF('day', DATE_ADD('day', 1, NOW()), NOW())
    , DATE_DIFF('hour', DATE_ADD('hour', 1, NOW()), NOW())
    , DATE_DIFF('minute', DATE_ADD('minute', 1, NOW()), NOW())
    , DATE_DIFF('second', DATE_ADD('second', 1, NOW()), NOW())
    , DATE_DIFF('millisecond', DATE_ADD('millisecond', 1, NOW()), NOW())
;
```

04

함수

날짜형 데이터의 데이터 타입 변경 (date -> string)

결과 (1)

복사 | 결과 다운로드

행 검색

1

#	_col0	_col1	_col2	_col3	_col4	_col5	_col6
1	2024-06-27 19:40:38.792 Asia/Seoul	2024-06-27	20240627	202406	19:40:38	07:40:38 PM	19:40:38

```
SELECT NOW()
    , DATE_FORMAT(NOW(), '%Y-%m-%d')
    , DATE_FORMAT(NOW(), '%Y%m%d')
    , DATE_FORMAT(NOW(), '%Y%m')
    , DATE_FORMAT(NOW(), '%H:%i:%s')
    , DATE_FORMAT(NOW(), '%r')
    , DATE_FORMAT(NOW(), '%T')
;

SELECT NOW()
    , CONCAT(CURRENT_DATE, CURRENT_TIME)
    , CONCAT(DATE_FORMAT(NOW(), '%Y-%m-%d'), DATE_FORMAT(NOW(),
'%H:%i:%s'))
;
```

04

함수

날짜형 데이터의 데이터 타입 변경 (string -> date)

```
SELECT DATE_PARSE('2024-03-27', '%Y-%m-%d');
SELECT DATE_PARSE('23:58:59', '%H:%i:%s');
SELECT DATE_PARSE('2024-06-28 15:23:23', '%Y-%m-%d %H:%i:%s');
```

04

함수

날짜형 데이터의 데이터 타입 변경 (cast)

```
SELECT NOW()
    , CAST(DATE_FORMAT(NOW(), '%Y-%m-%d %H:%i:%s') AS VARCHAR)
    , CAST(NOW() AS VARCHAR)
    , CAST(CAST(NOW() AS VARCHAR) AS TIMESTAMP)
;
```

04

함수

실습 5

e_media 테이블 확인 후 함수 사용해보기 !

proc_ymd 의 날짜가
해당 월의 마지막 날이라면 "Y",
해당 월의 마지막 날이 아니라면 "N"으로
표시되는 "last_day_yn" 컬럼을 만들고,
2024년 3월에 해당하는 proc_ymd와
last_day_yn 컬럼을
proc_ymd 오름차순으로 출력하세요.

04

함수

실습 5

e_media 테이블 확인 후 함수 사용해보기 !

```
SELECT DISTINCT
    "proc_ymd",
    -- DATE_PARSE("proc_ymd", '%Y%m%d'),
    -- DATE_ADD('month', 1, DATE_PARSE("proc_ymd", '%Y%m%d')),
    -- DATE_TRUNC('month', DATE_ADD('month', 1, DATE_PARSE("proc_ymd", '%Y%m%d'))),
    -- DATE_ADD('day', -1, DATE_TRUNC('month', DATE_ADD('month', 1, DATE_PARSE("proc_ymd", '%Y%m%d')))),
    -- DATE_FORMAT(DATE_ADD('day', -1, DATE_TRUNC('month', DATE_ADD('month', 1, DATE_PARSE("proc_ymd", '%Y%m%d')))), '%Y%m%d'),
    IF("proc_ymd" = DATE_FORMAT(DATE_ADD('day', -1, DATE_TRUNC('month', DATE_ADD('month', 1, DATE_PARSE("proc_ymd", '%Y%m%d')))), '%Y%m%d'), 'Y', 'N') AS "last_day_yn"
FROM "learning_analytics"."e_media"
WHERE 1=1
    AND "proc_ym" = '202403'
ORDER BY "proc_ymd" ASC
;
```

04

함수

윈도우 함수

GROUP BY (그룹화) 와 유사하게,
특정한 그룹에 대해 집계를 하는 함수
GROUP BY 가 테이블의 형태를
변화시키는 것과 달리,
윈도우 함수는 원본 그대로의 테이블에서
집계를 수행한다.

윈도우 함수 – 순위

RANK

- 행의 값이 같으면 같은 순위 부여
- 그 다음 값은 동일 값이 나온 숫자만큼 건너 뜀
- 예) 학교 등의 석차

DENSE_RANK

- 행의 값이 같으면 같은 순위 부여
- 같은 값은 하나의 건수로 취급
- 예) 올림픽 등의 순위

윈도우 함수 – 순위

ROW_NUMBER

- 같은 값이 나오더라도 고유한 값을 따로 부여함

04

함수

윈도우함수

```
SELECT *
FROM "learning_analytics"."e_learning_action"
WHERE 1=1
    AND "proc_ymd" = '20211031'
    AND "userid" = '6110dd0d-f737-49e6-ad22-24ce3be004af'
ORDER BY "learning_seq" ASC, "learning_action_seq" ASC
;
```

```
SELECT
    "proc_ymd", "userid", "mcode", "lecture_type",
    ROW_NUMBER() OVER(PARTITION BY "proc_ym", "userid", "mcode", "lecture_type" ORDER BY "eventtime_kst") AS "_ROW_NUMBER",
    RANK() OVER(PARTITION BY "proc_ym", "userid", "mcode", "lecture_type" ORDER BY "eventtime_kst") AS "_RANK",
    DENSE_RANK() OVER(PARTITION BY "proc_ym", "userid", "mcode", "lecture_type" ORDER BY "eventtime_kst") AS "_DENSE_RANK",
    "eventtime_kst"
FROM "learning_analytics"."e_learning_action"
WHERE 1=1
    AND "proc_ymd" = '20211031'
    AND "userid" = '6110dd0d-f737-49e6-ad22-24ce3be004af'
ORDER BY "mcode" ASC, "lecture_type" ASC, "_ROW_NUMBER" ASC
;
```

윈도우 함수 – 순위

_ROW_NUMBER	_RANK	_DENSE_RANK	eventtime_kst
1	1	1	2021-10-31 18:24:32.795
2	2	2	2021-10-31 18:24:32.806
3	3	3	2021-10-31 18:24:42.412
4	4	4	2021-10-31 18:24:53.544
5	5	5	2021-10-31 18:24:53.691
6	6	6	2021-10-31 18:24:54.709
7	6	6	2021-10-31 18:24:54.709
8	6	6	2021-10-31 18:24:54.709
9	9	7	2021-10-31 18:24:56.751
10	10	8	2021-10-31 18:25:02.137
11	11	9	2021-10-31 18:25:02.301
12	12	10	2021-10-31 18:25:19.385
13	13	11	2021-10-31 18:25:19.796

윈도우함수-실습

```
SELECT *
FROM "learning_analytics"."e_point"
WHERE 1=1
    AND "proc_ym" = '202403'
    AND "point" > 0
LIMIT 50;
```

위 쿼리를 토대로 윈도우 함수를 이용하여
처리년월일, 회원고유 ID 별
point 가 많은 순서대로
RANK 와 DENSE_RANK 컬럼을 만들어 봅시다.

04

함수

윈도우함수-더보기

```
SELECT
    *,
    SUM("sum_point") OVER (PARTITION BY "userid"
                ORDER BY "proc_ymd" ASC
                ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS "test"
FROM (
    SELECT
        "proc_ymd", "userid",
        SUM("point") AS "sum_point"
    FROM "learning_analytics"."e_point"
    WHERE 1=1
        AND "point" > 0
        AND "proc_ym" = '202403'
    GROUP BY "proc_ymd", "userid"
    )
```

CONTENTS

05

집합

SELECT 쿼리의 작성 순서 2

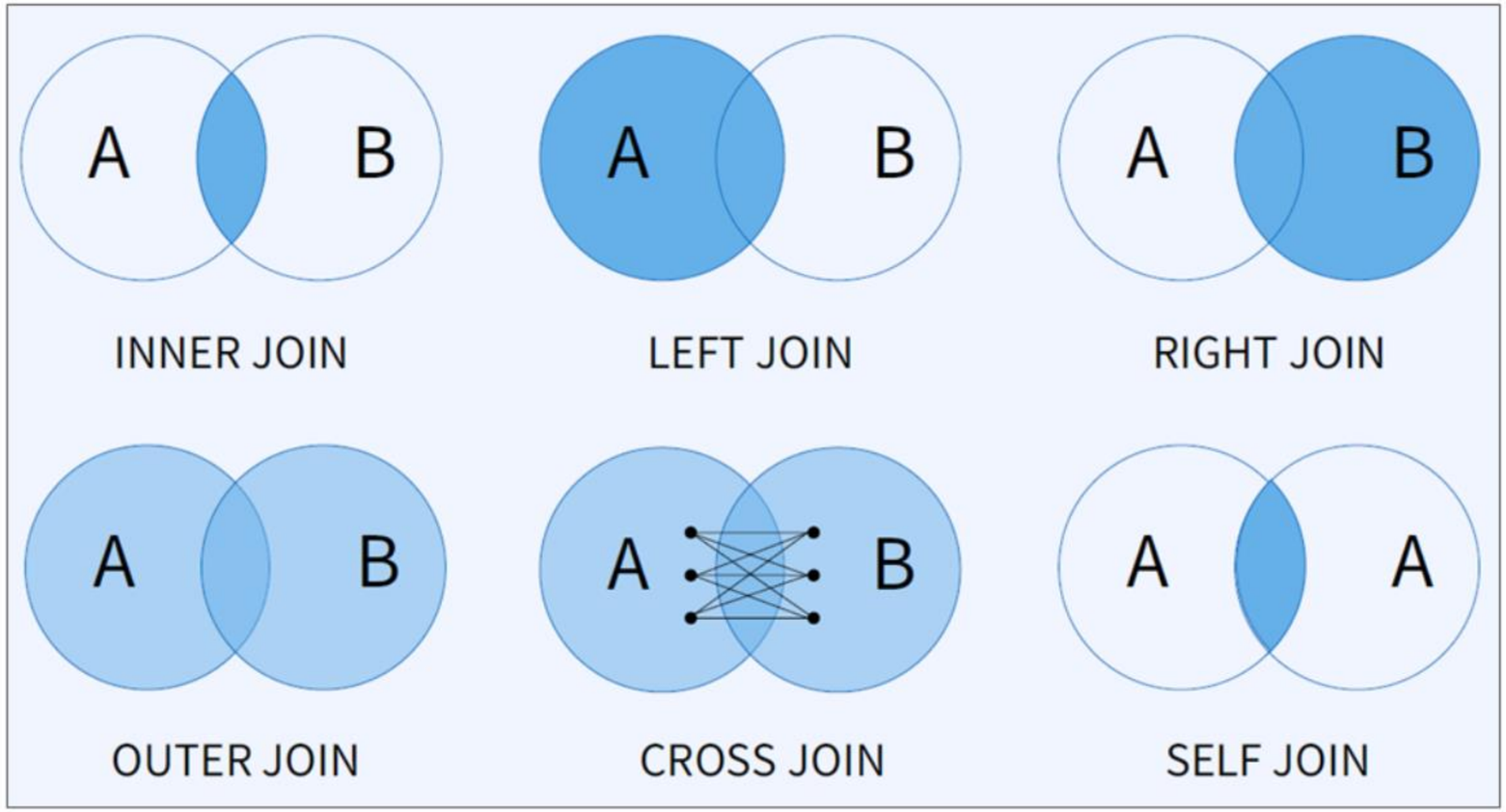
```
FROM table_name AS a
    JOIN table_name AS b
    ON a.column_name = b.column_name
```

SELECT 쿼리의 작성 순서	SELECT 쿼리의 실행 순서
SELECT	FROM
FROM	ON
JOIN	JOIN
ON	WHERE
WHERE	GROUP BY
GROUP BY	(HAVING)
(HAVING)	SELECT
ORDER BY	ORDER BY

05

집합

JOIN의 종류



05

집합

JOIN

이름	직급
김지훈	A
김지훈	B
엄상민	C
이윤재	D
황민정	D
왕지수	G

직급	급여
A	13000
B	12000
C	11000
D	10000
E	9000
F	8000

위의 두 테이블의 직급 컬럼으로
JOIN을 시도해 봅시다

05

집합

INNER JOIN

이름	직급
김지훈	A
김지훈	B
엄상민	C
이윤재	D
황민정	D
왕지수	G

직급	급여
A	13000
B	12000
C	11000
D	10000
E	9000
F	8000

이름	직급	급여
김지훈	A	13000
김지훈	B	12000
엄상민	C	11000
이윤재	D	10000
황민정	D	10000

05

집합

LEFT (OUTER) JOIN

이름	직급
김지훈	A
김지훈	B
엄상민	C
이윤재	D
황민정	D
왕지수	G

직급	급여
A	13000
B	12000
C	11000
D	10000
E	9000
F	8000

이름	직급	급여
김지훈	A	13000
김지훈	B	12000
엄상민	C	11000
이윤재	D	10000
황민정	D	10000
왕지수	G	

05

집합

RIGHT (OUTER) JOIN

이름	직급
김지훈	A
김지훈	B
엄상민	C
이윤재	D
황민정	D
왕지수	G

직급	급여
A	13000
B	12000
C	11000
D	10000
E	9000
F	8000

이름	직급	급여
김지훈	A	13000
김지훈	B	12000
엄상민	C	11000
이윤재	D	10000
황민정	D	10000
	E	9000
	F	8000

05

집합

FULL OUTER JOIN

이름	직급
김지훈	A
김지훈	B
엄상민	C
이윤재	D
황민정	D
왕지수	G

직급	급여
A	13000
B	12000
C	11000
D	10000
E	9000
F	8000

이름	직급	급여
김지훈	A	13000
김지훈	B	12000
엄상민	C	11000
이윤재	D	10000
황민정	D	10000
	E	9000
	F	8000
왕지수	G	

05

집합

CROSS JOIN

상의
흰 티셔츠
검은 티셔츠
노란 후드티

하의
긴 청바지
흰색 반바지

상의	하의
흰 티셔츠	긴 청바지
검은 티셔츠	긴 청바지
노란 후드티	긴 청바지
흰 티셔츠	흰색 반바지
검은 티셔츠	흰색 반바지
노란 후드티	흰색 반바지

05

집합

SELF JOIN?

id	first_name	last_name	salary	manager_id
1	John	Watson	7550	NULL
2	Anne	Brown	3500	1
3	James	Black	3000	1
4	Scarlett	Miller	2500	3
5	Ethan	Davis	1200	3
6	Jacob	Smith	2000	3

```
SELECT
 e.id,
 e.first_name,
 e.last_name,
 e.salary,
 m.first_name AS boss_first_name
 m.last_name AS boss_last_name
FROM employee AS e
INNER JOIN employee AS m
    ON e.manager_id = m.id;
```

05

집합

함께해 보기 !

e_assessment 테이블과
e_assessment_detail 테이블을
확인해 봅시다.
두 테이블은 "proc_ymd", "userid", "mcode" 컬럼으로
JOIN 을 할 수 있습니다.
두 테이블을 위 컬럼들로 INNER JOIN 하고,
"proc_ymd" 컬럼 값이 '20230101' 인 행들만
출력해 봅시다.

05

집합

두 개 이상의 데이터를 합하는 다른 방법

시놉시스

```
[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT ] select_expression [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition ]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
[ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST] [, ...] ]
[ OFFSET count [ ROW | ROWS ] ]
[ LIMIT [ count | ALL ] ]
```

UNION ALL

쿼리의 결과를 합한다

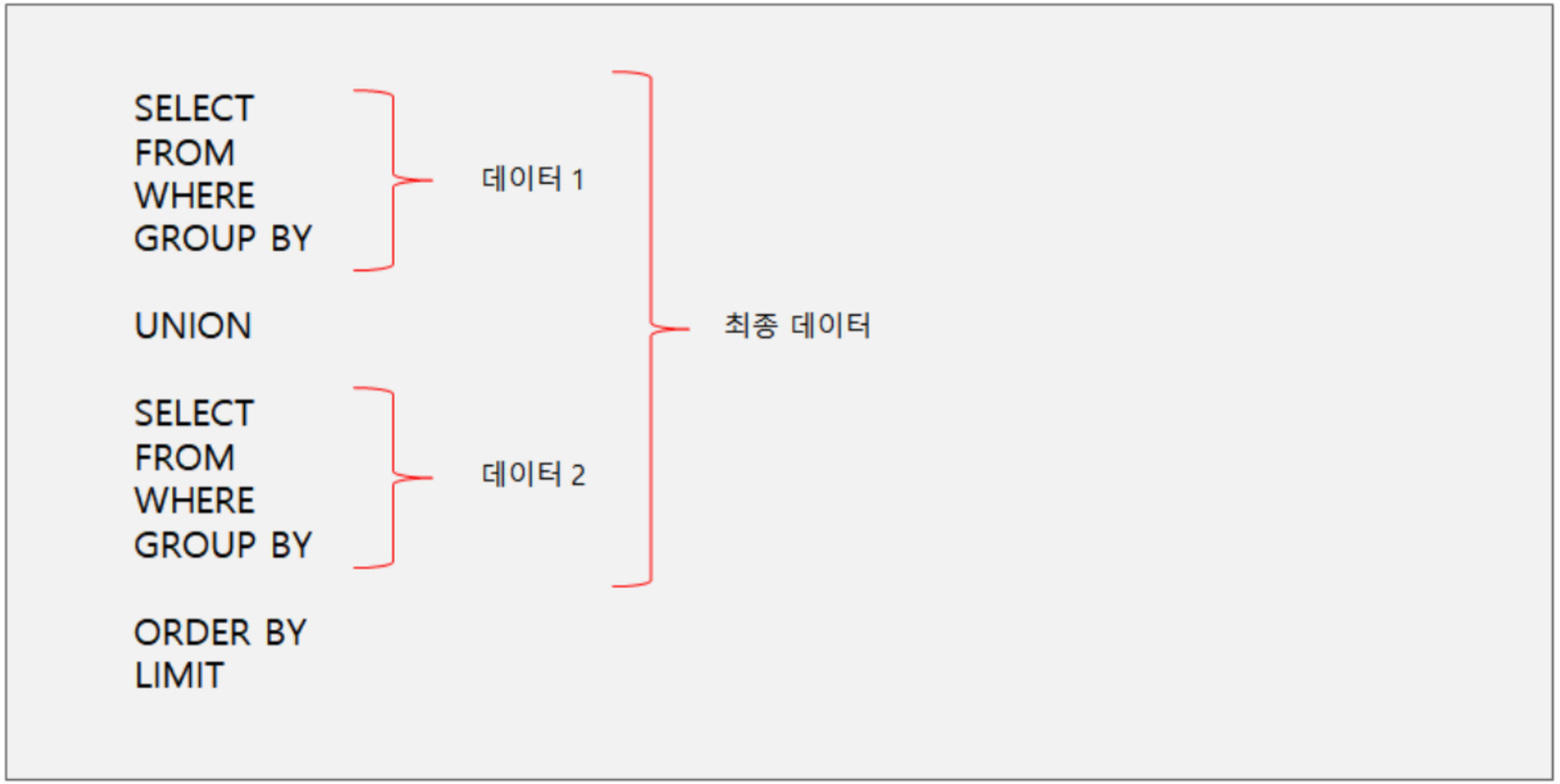
UNION DISTINCT

쿼리의 결과를 합한다
중복을 제거한다

05

집합

UNION


SELECT
FROM
WHERE
GROUP BY
데이터 1
UNION
SELECT
FROM
WHERE
GROUP BY
데이터 2
최종 데이터
ORDER BY
LIMIT

05

집합

UNION

```
SELECT 1,2,3
UNION
SELECT 4,5,6
;
```

```
SELECT 1,2,3
UNION ALL
SELECT 1,2,3
;
```

```
SELECT 1,2,3
UNION
SELECT 1,2,3
;
```

CONTENTS

06

WITH의 활용

변수 선언

WITH

- SELECT 문 전에 가상의 테이블을 만들어 활용하는 방법
- WITH 문을 잘 사용하면 가독성이 높게 쿼리를 짤 수 있음

```
WITH _temp1 AS (
        VALUES (CHAR '202208')
    )
    , _temp2 (yyyy, mm) AS (
        VALUES (CHAR '2022', CHAR '08')
    )
```

06

WITH의 활용

테이블 선언

```
WITH _temp1 AS (
        VALUES (CHAR '202208')
    )
    , _temp2 ("yyyy", "mm") AS (
        VALUES (VARCHAR '2022', VARCHAR '08')
    )
    , _temp3 AS (
        SELECT *
        FROM "learning_analytics"."e_assessment"
        WHERE CONCAT("yyyy", "mm") = '202208'
    )
    , _temp4 AS (
        SELECT *
        FROM "learning_analytics"."e_assessment"
        WHERE CONCAT("yyyy", "mm") = (SELECT * FROM _temp1)
    )
    , _temp5 AS (
        SELECT *
        FROM "learning_analytics"."e_assessment"
        WHERE 1=1
            AND "yyyy" = (SELECT "yyyy" FROM _temp2)
            AND "mm" = (SELECT "mm" FROM _temp2)
    )
    , _temp6 ("년월일", "아이디", "mcode", "점수") AS (
        SELECT "proc_ymd", "userid", "mcode", "score"
        FROM _temp5
    )

SELECT *
FROM _temp6
LIMIT 10;
```

과제 내용

평가 기준
1. 쿼리가 정상적으로 실행 가능한가?
2. 행 수가 답과 일치하는가?
3. 사용자 기능이 구현되었는가?
4. 적절한 컬럼을 선택하였는가?

text_biz_dw 데이터베이스를 활용하여
다음 요구사항을 만족하는 SQL을 작성하세요.
2022년 분기별 (총 4분기) 콘텐츠 이용
실태 조사를 진행하고자 합니다.
초등 3학년, 4학년, 5학년, 6학년을 대상으로 하는 콘텐츠 중
영상강의 + 문제풀이가 함께 서비스되는 콘텐츠를 대상으로 하여,

1. 콘텐츠 별 학습을 진행한 학생 수
2. 콘텐츠 별 학습을 진행한 학생의 학년 평균 (학생학년별 평균이 아니라 학년값의 평균)
3. 콘텐츠 별 학습 시간
4. 콘텐츠 별
   평가문항 평균 갯수, 정답 문항 평균 갯수, 평가점수 평균

(평균 값은 소수 둘째자리까지 표현 해 주세요.)
을 확인할 수 있는 쿼리를 전달해 주세요.
이 때, WITH 문을 사용하여
사용자의 분기 입력을 통해 분기 별 데이터를 조회할 수 있는 방법을 만들어 주세요.
사용 테이블 : e_media, e_content_meta, e_test, e_study, e_member

CHUNJAE EDUCATION

고맙습니다