코멘토 실무PT 클래스

개발자 없이 시작하는 데이터 분석 - 비개발자를 위한 SQL

CONTENTS

1주차 2주차 3주차 **4주차** 5주차 6주차

22.10.30(일) 20:00~23:00

4주차
고객의 첫 주문이 알고 싶어요!

01 3주차 세션 회고 및 과제 피드백

02 조건문(CASE), UNION, EXCEPT

03 현업에서 자주 사용하는 함수 설명
- 내재함수 (TYPE, DESCRIBE, EXPLAIN)
- 날짜함수 (날짜 변환(e.g. 요일추출), 날짜 연산)
- 문자함수 (숫자를 문자로)
- 수학함수 (통계, 제곱)

04 고급함수 맛보기
- WINDOW 분석함수 (리텐션, 코호트)
- 정규표현식(휴대폰번호 추출)
- UDF(사용자 정의 함수)

05 실시간 Q&A

과제 함수 & WINODW 실습
- 유저의 첫 구매를 구하고, 주차별 리텐션율을 구해보기

Chapter.

1. 지난시간

CONTENTS

22.10.23(일) 20:00~23:00

3주차
데이터들의 "조합"으로 뽑아봅시다

01 2주차 세션 회고 및 과제 피드백

02 JOIN
- JOIN의 종류, 결합키 찾기
- ERD 통한 테이블 관계 확인

03 SUB QUERY
- SUB QUERY의 종류(WITH절), 사용시 유의사항

04 현직자 이야기 : SQL을 잘 읽고 쓰는 법 (코드컨벤션 설명)

05 VSCODE 설치 및 소개

06 실시간 Q&A

과제 JOIN, SUB QUERY 활용하기
- 공공데이터(자전거대여BikeShare) 데이터 추출 연습 진행

Chapter.

1. 과제피드백

데일리 퀴즈

Q. (문제)
- 대여 정류소별 따릉이를 빌려간 횟수를 구하는 쿼리를 짜주세요.
- 이때, 대여 정류소이름(name)과 소속 지자체(local)를 함께 할 수 있도록 넣어주세요.
- 정류소ID 오름차순으로 정렬해주세요.

(테이블)
1. station : 정류소 테이블
  - station_id : 정류소ID(연결고리)
  - name : 정류소 이름
  - local : 소속 지자체
2. rental_history : 대여이력 테이블
  - rent_station_id : 대여 정류소ID(연결고리)

—-
A.

```
SELECT
  rh.rent_station_id,
  s.name,
  s.local,
  COUNT(*)
FROM rental_history rh
  JOIN station s
    ON rh.rent_station_id = s.station_id
GROUP BY
  rh.rent_station_id,
```

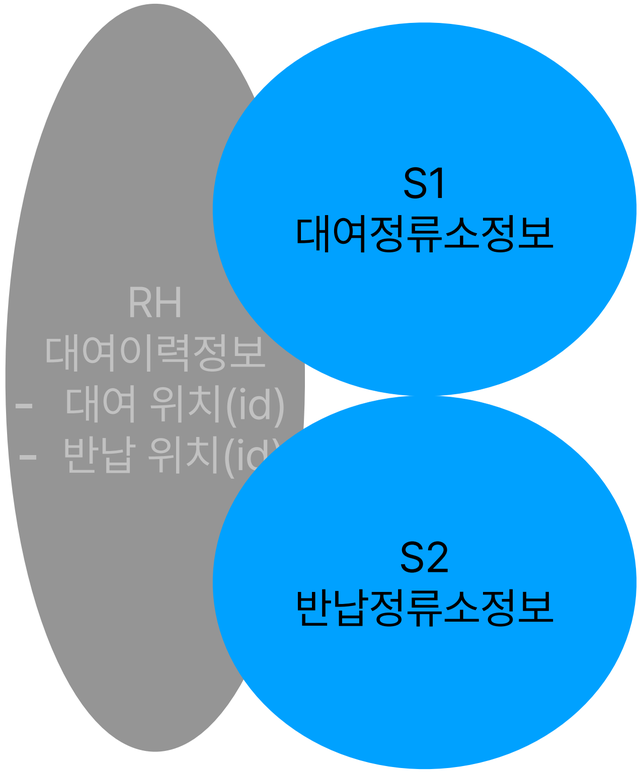
	rent_station_id	name	local	COUNT(*)
1	101	(구)합정동 주민센터	마포구	1390
2	102	망원역 1번출구 앞	마포구	5421
3	103	망원역 2번출구 앞	마포구	3689
4	104	합정역 1번출구 앞	마포구	3525
5	105	합정역 5번출구 앞	마포구	2145
6	106	합정역 7번출구 앞	마포구	6532
7	107	신한은행 서교동금융센터점 앞	마포구	4278
8	108	서교동 사거리	마포구	3062
9	109	제일빌딩 앞	마포구	3311
10	111	상수역 2번출구 앞	마포구	3064
11	112	극동방송국 앞	마포구	3085
12	113	홍대입구역 2번출구 앞	마포구	10148
13	114	홍대입구역 8번출구 앞	마포구	5322
14	115	마스타 빌딩 앞	서대문구	2748
15	116	일진아이윌아파트 옆	서대문구	3193
16	117	홍은사거리	서대문구	3531
17	118	광흥창역 2번출구 앞	마포구	2833

Page 1 of 22

데일리 퀴즈

조인을 여러개를 쓰는 케이스
— >.JOIN을 중복해서 쓰는 케이스를 한번 보여드릴게요.

```
SELECT
  DISTINCT
  — 1. 대여정류소
  rh.rent_station_id,
  s1.name as rent_station_name,
  s1.lat as rent_lat,
  s1.lng as rent_lng,
  — 2. 반납정류소
  rh.return_station_id,
  s2.name as return_station_name,
  s2.lat as return_lat,
  s2.lng as return_lng
FROM rental_history rh
  JOIN station s1 — 1. 대여정류소
    ON rh.rent_station_id = s1.station_id — 대여정류소의 정보
  JOIN station s2 — 2. 반납정류소
    ON rh.return_station_id = s2.station_id — 반납정류소의 정보
WHERE rh.rent_station_id = 101
```



연결고리는 하나의 관계로만 가능
—
RH ~ s1 연결고리 : **대여** 정류소의 ID
RH ~ s2 연결고리 : **반납** 정류소의 ID

Chapter.별도

따릉이 자전거 이용 현황 (from ~ to) 지리 데이터 시각화


kepler.gl
2.5.5
Base map
Map Style
Dark
Map Layers
Label
Road
Border
Building
Water
Land
3d Building
3D Building Color
+ Add Map Style
Namyangju
Dobong 2-dong
KOREA MILITARY ACADEMY
Guri
SEOOREUNG ROYAL TOMBS
BUGAKSAN MOUNTAIN
JUNGNANG SEWAGE TREATMENT CENTER
Seoul
YONSEI UNIVERSITY SINCHON CAMPUS
NAMSAN PARK
NOEUL PARK
HANEUL PARK
USAG YONGSAN
Heukseok-dong
Seorae Village
EN
Basemap by: mapbox
© kepler.gl | © Mapbox | © OpenStreetMap | Improve this map

```
SELECT
  rh.bike_id,
  rh.distance,
  rh.return_at,
  rh.rent_station_id,
  s1.lat as rent_lat,
  s1.lng as rent_lng,
  rh.return_station_id,
  s2.lat as return_lat,
  s2.lng as return_lng
FROM rental_history rh
  LEFT JOIN station s1
    ON rh.rent_station_id = s1.station_id
  LEFT JOIN station s2
    ON rh.return_station_id = s2.station_id
WHERE 1=1
AND s1.local = '성동구'
LIMIT 4999
```

1. 왼쪽 쿼리를 solvesql.com에서 실행.
2. 실행한 결과를 CSV로 저장
3. https://kepler.gl/ 접속
4. 다운 받았던 CSV를 업로드하기.
5. 마음껏 데이터를 탐색해봅시다.

없어진 기록찾기

없어진 기록 찾기

dark light | sublime vim emacs | MySQL

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID` , `ANIMAL_TYPE` , `DATETIME` , `INTAKE_CONDITION` , `NAME` , `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
INTAKE_CONDITION	VARCHAR(N)	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_INTAKE	VARCHAR(N)	FALSE

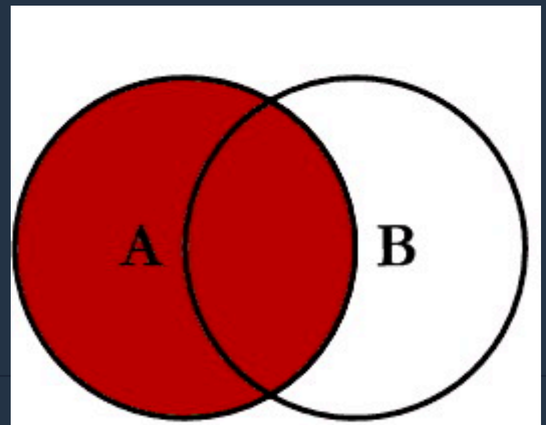
`ANIMAL_OUTS` 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다. `ANIMAL_OUTS` 테이블 구조는 다음과 같으며, `ANIMAL_ID` , `ANIMAL_TYPE` , `DATETIME` , `NAME` , `SEX_UPON_OUTCOME` 는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다. `ANIMAL_OUTS` 테이블의 `ANIMAL_ID` 는 `ANIMAL_INS` 의 `ANIMAL_ID` 의 외래 키입니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE

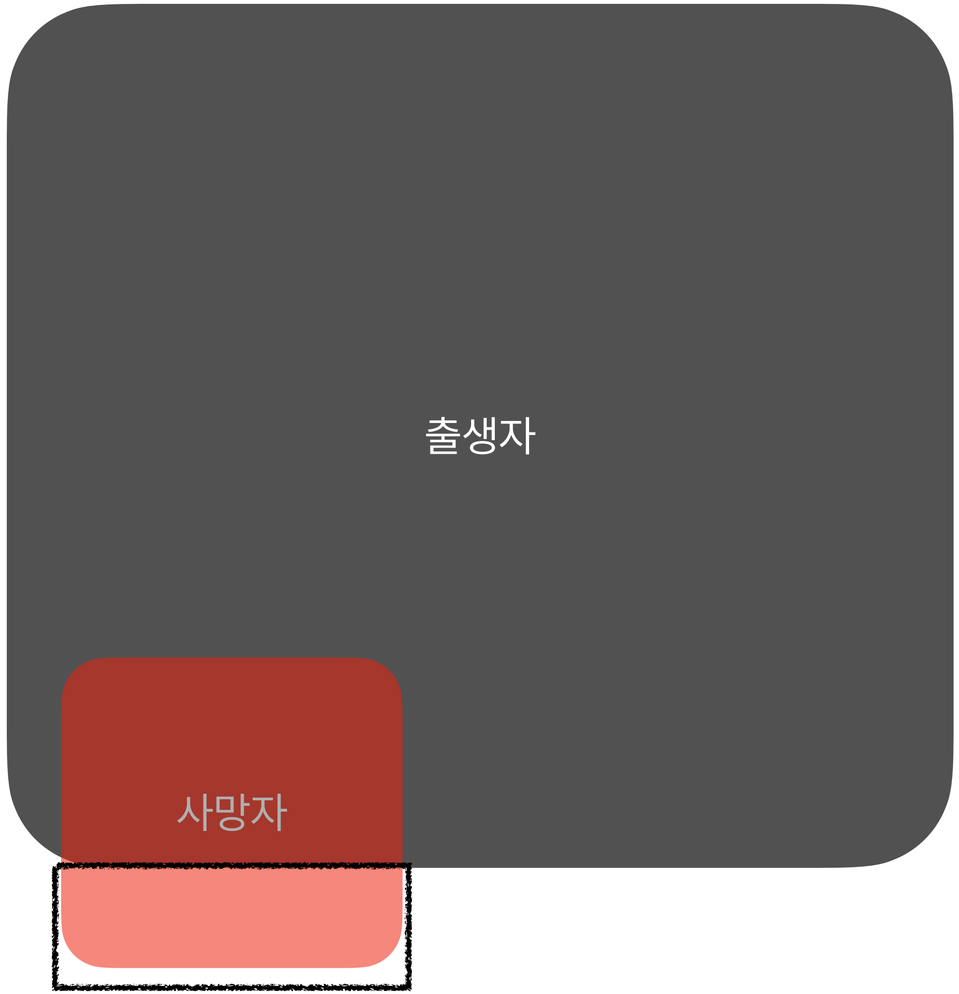
solution.sql

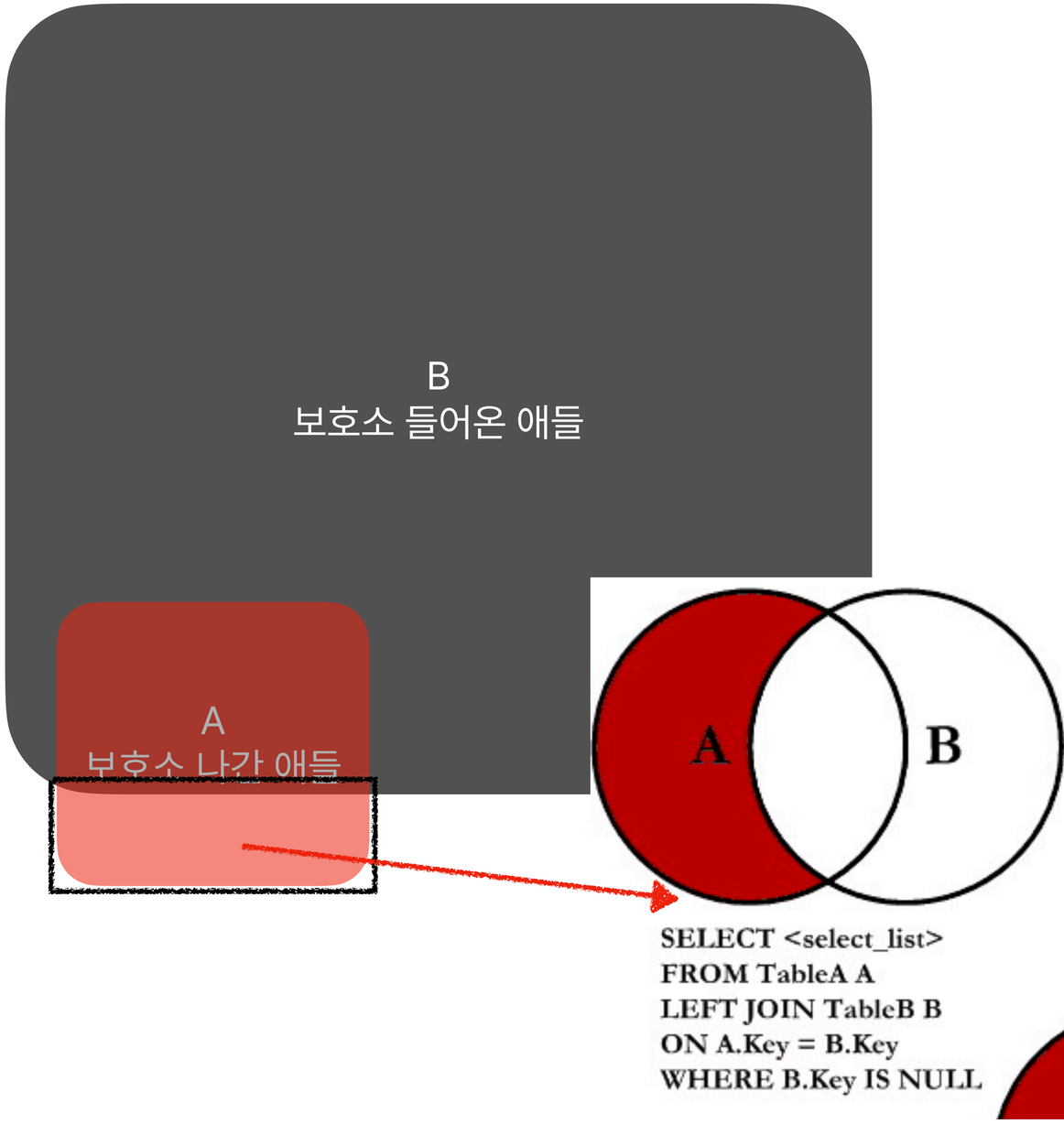
```sql
# 천재지변으로 인해 일부 데이터가 유실되었습니다.
# 입양을 간 기록은 있는데, 보호소에 들어온 기록이 없는 동물의 ID와 이름을 ID 순으로 조회하는 SQL문을 작성해주세요.

SELECT
    OUTS.ANIMAL_ID AS out_animal_id, -- 나간 동물ID
    OUTS.NAME
FROM ANIMAL_OUTS OUTS -- 있고.(얘가 큰 상황)
    LEFT JOIN ANIMAL_INS INS -- 데이터가 일부 없음.
        ON OUTS.ANIMAL_ID = INS.ANIMAL_ID
WHERE INS.ANIMAL_ID IS NULL -- 들어온 동물ID 값이 없는 경우.

# 같은 결과. RIGHT 조인으로 표현
SELECT
    OUTS.ANIMAL_ID,
    OUTS.NAME
FROM ANIMAL_INS INS
    RIGHT JOIN ANIMAL_OUTS OUTS
        ON INS.ANIMAL_ID = OUTS.ANIMAL_ID
WHERE INS.ANIMAL_ID IS NULL
```

실행 결과

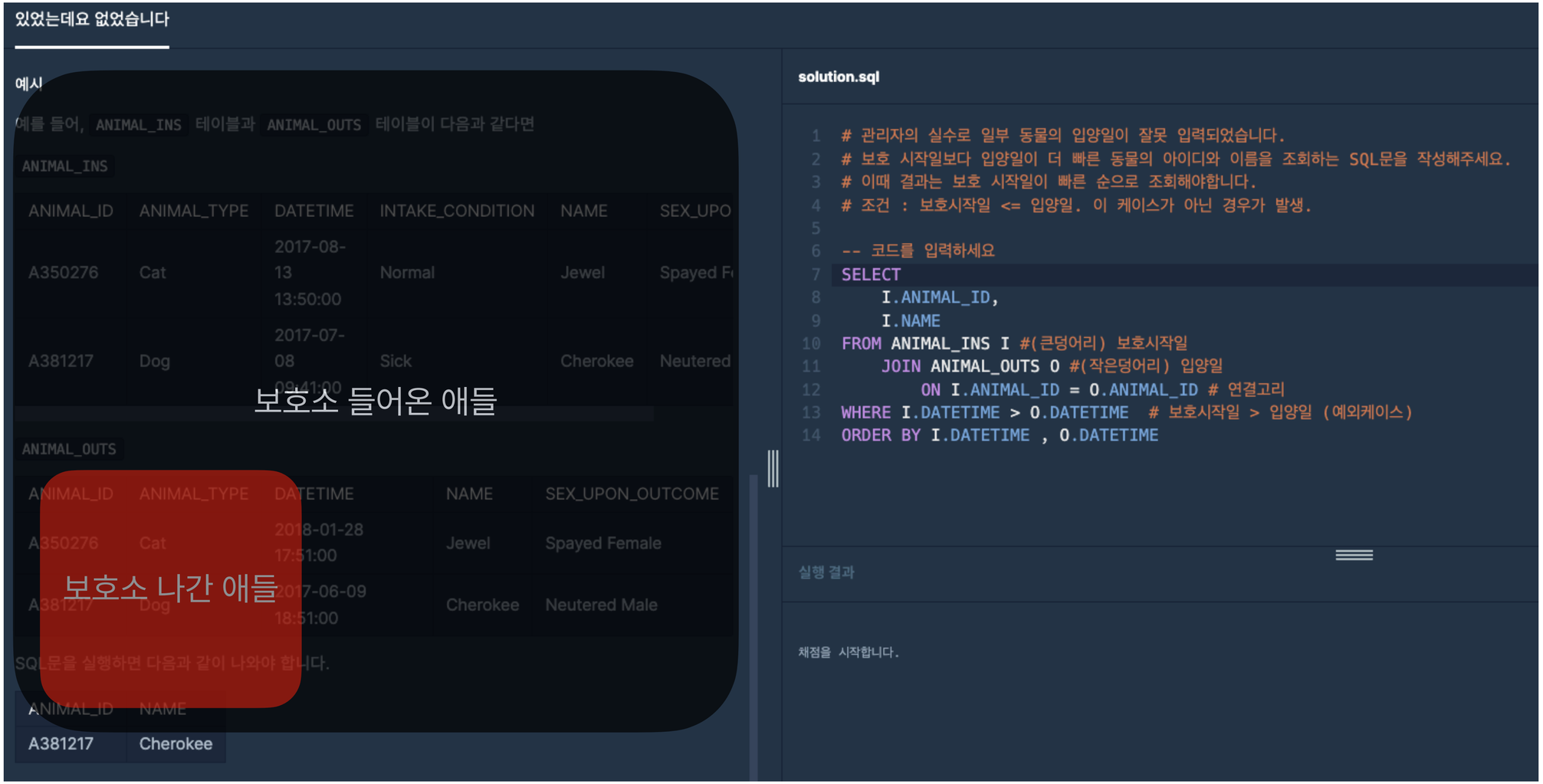


SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key

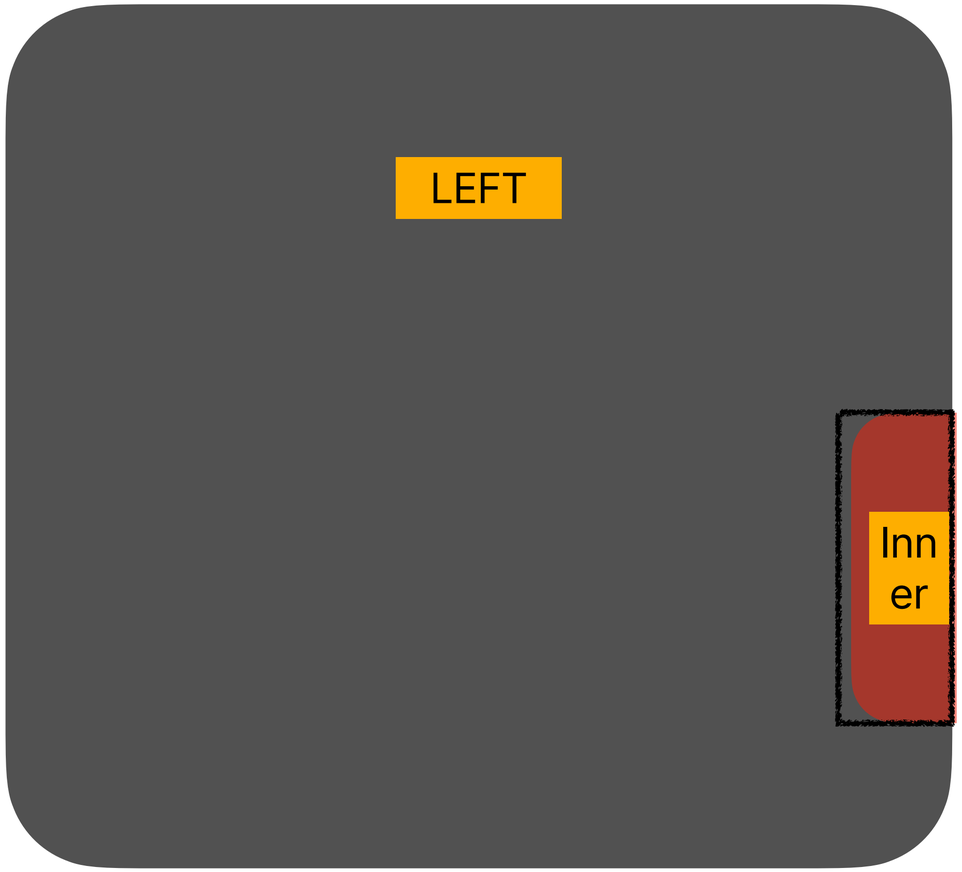

출생자
사망자

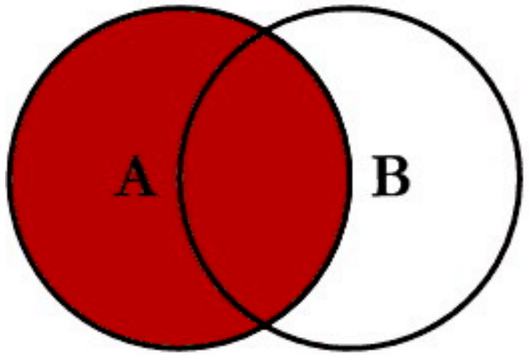

B
보호소 들어온 애들
A
보호소 나간 애들
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL

있었는데요 없었습니다.


있었는데요 없었습니다
예시
예를 들어, ANIMAL_INS 테이블과 ANIMAL_OUTS 테이블이 다음과 같다면
ANIMAL_INS
ANIMAL_ID ANIMAL_TYPE DATETIME INTAKE_CONDITION NAME SEX_UPO
A350276 Cat 2017-08-13 13:50:00 Normal Jewel Spayed F
A381217 Dog 2017-07-08 Sick Cherokee Neutered
보호소 들어온 애들
ANIMAL_OUTS
ANIMAL_ID ANIMAL_TYPE DATETIME NAME SEX_UPON_OUTCOME
A350276 Cat 2018-01-28 17:51:00 Jewel Spayed Female
A381217 Dog 2017-06-09 18:51:00 Cherokee Neutered Male
보호소 나간 애들
SQL문을 실행하면 다음과 같이 나와야 합니다.
ANIMAL_ID NAME
A381217 Cherokee
solution.sql
관리자의 실수로 일부 동물의 입양일이 잘못 입력되었습니다.
보호 시작일보다 입양일이 더 빠른 동물의 아이디와 이름을 조회하는 SQL문을 작성해주세요.
이때 결과는 보호 시작일이 빠른 순으로 조회해야합니다.
조건 : 보호시작일 <= 입양일. 이 케이스가 아닌 경우가 발생.
-- 코드를 입력하세요
SELECT
I.ANIMAL_ID,
I.NAME
FROM ANIMAL_INS I #(큰덩어리) 보호시작일
JOIN ANIMAL_OUTS O #(작은덩어리) 입양일
ON I.ANIMAL_ID = O.ANIMAL_ID # 연결고리
WHERE I.DATETIME > O.DATETIME # 보호시작일 > 입양일 (예외케이스)
ORDER BY I.DATETIME , O.DATETIME
실행 결과
채점을 시작합니다.

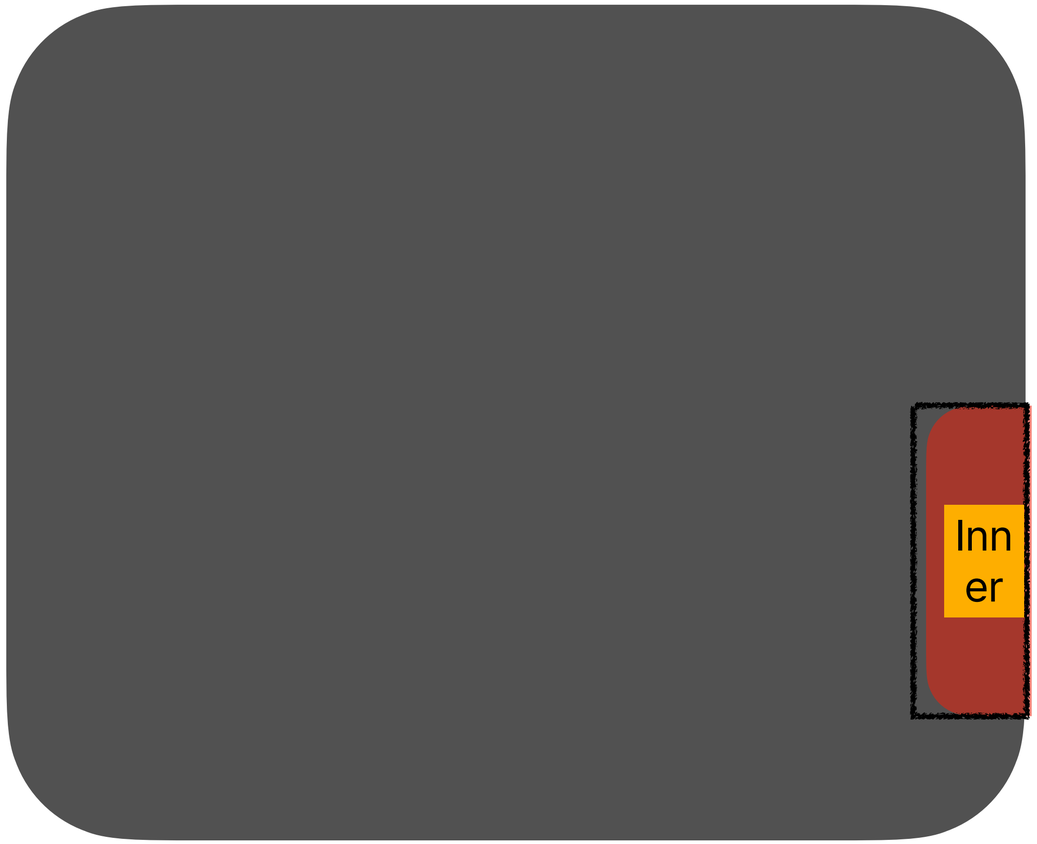
INNER(교집합)

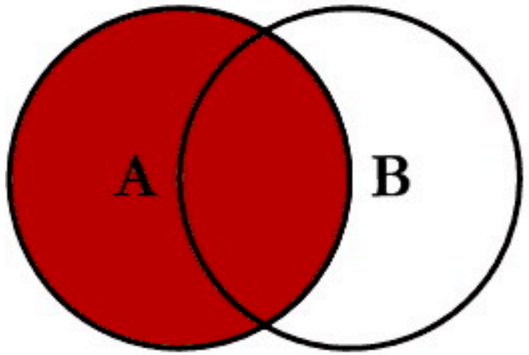





```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

INNER(교집합)






```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

오랜 기간 보호한 동물(1)

오랜 기간 보호한 동물(1)

dark light sublime

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID` , `ANIMAL_TYPE` , `DATETIME` , `INTAKE_CONDITION` , `NAME` , `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
INTAKE_CONDITION	VARCHAR(N)	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_INTAKE	VARCHAR(N)	FALSE

`ANIMAL_OUTS` 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다. `ANIMAL_OUTS` 테이블 구조는 다음과 같으며, `ANIMAL_ID` , `ANIMAL_TYPE` , `DATETIME` , `NAME` , `SEX_UPON_OUTCOME` 는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다. `ANIMAL_OUTS` 테이블의 `ANIMAL_ID` 는 `ANIMAL_INS` 의 `ANIMAL_ID` 의 외래 키입니다.

solution.sql

```sql
# 조건WHERE : 아직 입양을 못 간 동물(OUT이 없다) 중, "가장 오래 보호소"(MIN_DATETIME)에 있었던
# 조회SELECT : "동물 3마리의 이름NAME"과 "보호 시작일INS.DATETIME"을 조회하는 SQL문을 작성해주세요.
# 순서ORDER : 이때 결과는 "보호 시작일"(INS.DATETIME ASC) 순으로 조회해야 합니다.
-- 코드를 입력하세요
SELECT
    # I.ANIMAL_ID, -- 고유한ID값.(UNIQUE)
    I.NAME, -- 중복이 가능.(jane, jane, jane.. 중복 되는 케이스.)
    I.DATETIME -- 가장 오래된 애들을 알 수 있는 정보.(가장 작은 값들만 뽑아야함.)
FROM ANIMAL_INS I -- (큰덩어리) 1.보호소들어온애들
    LEFT JOIN ANIMAL_OUTS O -- (작은덩어리) 2.입양간애들
        ON I.ANIMAL_ID = O.ANIMAL_ID -- 1,2의 연결고리.
WHERE O.ANIMAL_ID IS NULL -- 입양을 못 간 동물(1번에있는데, 2번에는없는애)
ORDER BY I.DATETIME ASC -- 보호시작일 순.
LIMIT 3
```

실행 결과

보호소에서 중성화한 동물

보호소에서 중성화한 동물

dark light | sublime vim emacs | MySQL

문제 설명

`ANIMAL_INS` 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다. `ANIMAL_INS` 테이블 구조는 다음과 같으며, `ANIMAL_ID` , `ANIMAL_TYPE` , `DATETIME` , `INTAKE_CONDITION` , `NAME` , `SEX_UPON_INTAKE` 는 각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

NAME	TYPE	NULLABLE
ANIMAL_ID	VARCHAR(N)	FALSE
ANIMAL_TYPE	VARCHAR(N)	FALSE
DATETIME	DATETIME	FALSE
INTAKE_CONDITION	VARCHAR(N)	FALSE
NAME	VARCHAR(N)	TRUE
SEX_UPON_INTAKE	VARCHAR(N)	FALSE

`ANIMAL_OUTS` 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다. `ANIMAL_OUTS` 테이블 구조는 다음과 같으며, `ANIMAL_ID` , `ANIMAL_TYPE` , `DATETIME` , `NAME` , `SEX_UPON_OUTCOME` 는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다. `ANIMAL_OUTS` 테이블의 `ANIMAL_ID` 는 `ANIMAL_INS` 의 `ANIMAL_ID` 의 외래 키입니다.

NAME	TYPE	NULLABLE

solution.sql

```sql
# 보호소에서 중성화 수술을 거친 동물 정보를 알아보려 합니다.
# 보호소에 "들어올" 당시에는 중성화 되지 않았지만,
# 보호소를 "나갈" 당시에는 중성화된
# 동물의 아이디와 생물 종, 이름을 조회하는 아이디 순으로 조회하는 SQL 문을 작성해주세요.
SELECT
    I.ANIMAL_ID,
    I.ANIMAL_TYPE,
    I.NAME
FROM ANIMAL_INS I -- (큰덩어리) 1. 보호소 들어온.
    LEFT JOIN ANIMAL_OUTS O -- (작은덩어리) 2. 보호소 나간.
        ON I.ANIMAL_ID = O.ANIMAL_ID
WHERE (I.SEX_UPON_INTAKE NOT IN ('Spayed Female', 'Neutered Male') -- 보호소에 "들어올" 당시에는 중성화 되지 않았지만,
        AND O.SEX_UPON_OUTCOME IN ('Spayed Female', 'Neutered Male'))-- 보호소를 "나갈" 당시에는 중성화된
ORDER BY I.ANIMAL_ID -- 아이디 순으로 조회하는
```

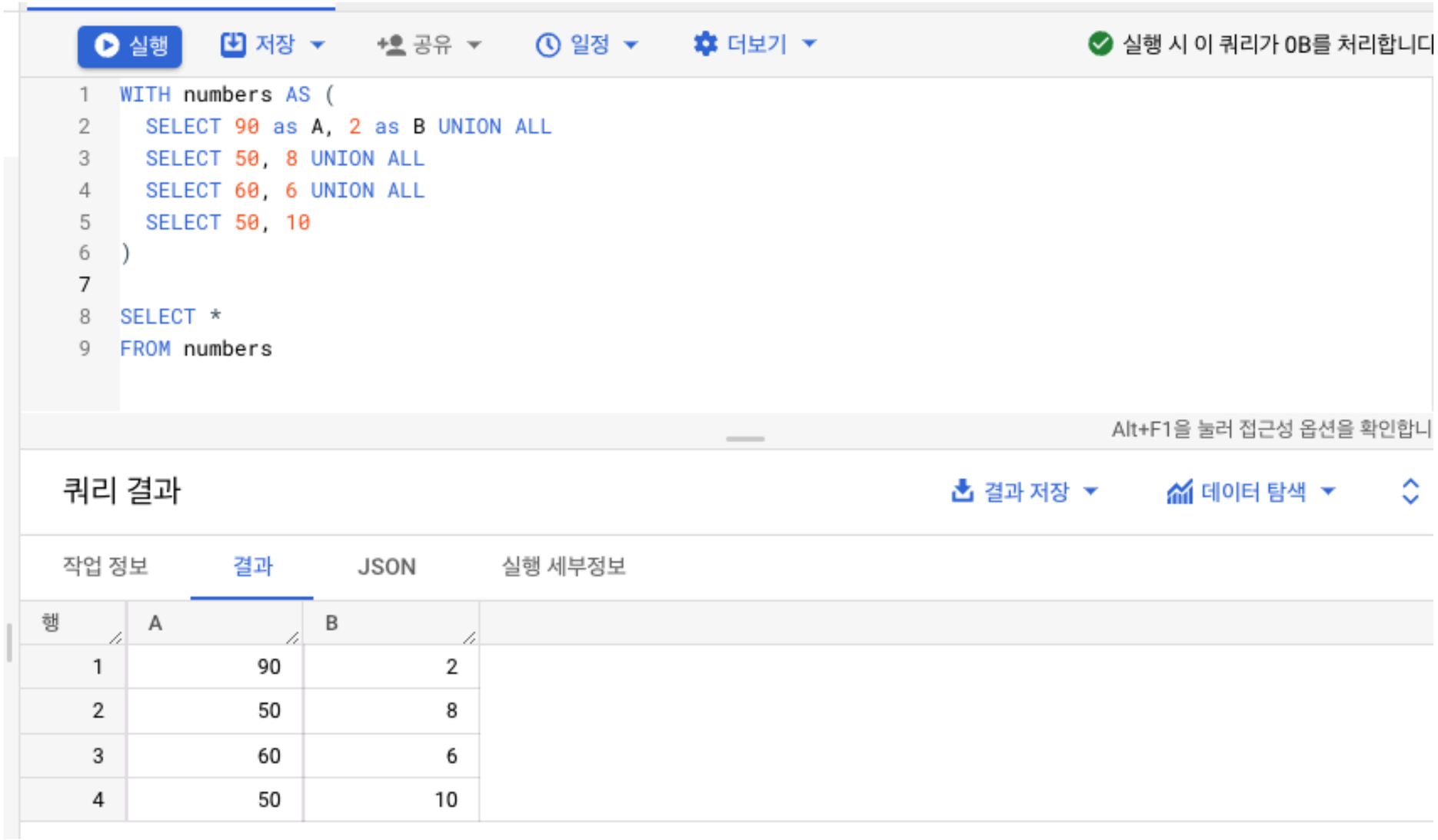
실행 결과

채점을 시작합니다.

Chapter.

테이블을 합치는 UNION

테이블을 합치기


실행
저장
공유
일정
더보기
실행 시 이 쿼리가 0B를 처리합니다
WITH numbers AS (
SELECT 90 as A, 2 as B UNION ALL
SELECT 50, 8 UNION ALL
SELECT 60, 6 UNION ALL
SELECT 50, 10
)
SELECT *
FROM numbers
Alt+F1을 눌러 접근성 옵션을 확인합니
쿼리 결과
결과 저장
데이터 탐색
작업 정보
결과
JSON
실행 세부정보
행
A
B
1
90
2
2
50
8
3
60
6
4
50
10

Chapter.

조건문 CASE

https://cloud.google.com/bigquery/docs/reference/standard-sql/conditional_expressions?hl=ko

Chapter.

함수Function (날짜, 숫자, 문자)

1. 변환함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/conversion_functions?hl=ko

2. 날짜함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/date_functions?hl=ko

3. 수학함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/mathematical_functions?hl=ko

4. 문자열 함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/string_functions?hl=ko

Chapter.

분석 함수Analytic(window) Function

탐색 함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/navigation_functions?hl=ko

윈도우 함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/window-function-calls?hl=ko

번호 지정 함수 : https://cloud.google.com/bigquery/docs/reference/standard-sql/numbering_functions

첫시작을 구해보기

```sql
SELECT
  bikeid,
  starttime,
  ROW_NUMBER() OVER(PARTITION BY bikeid ORDER BY starttime) AS sequence
FROM `bigquery-public-data.new_york_citibike.citibike_trips`
WHERE bikeid = 18447
ORDER BY 2
```

Alt+

쿼리 결과

결과 저장

작업 정보 | 결과 | JSON | 실행 세부정보

행	bikeid	starttime	sequence
1	18447	2013-07-02T12:52:00	1
2	18447	2013-07-02T13:44:15	2
3	18447	2013-07-02T13:54:22	3
4	18447	2013-07-02T14:14:06	4
5	18447	2013-07-03T08:08:59	5
6	18447	2013-07-03T18:02:30	6
7	18447	2013-07-03T18:16:00	7
8	18447	2013-07-03T18:32:22	8
9	18447	2013-07-03T18:39:43	9
10	18447	2013-07-03T19:03:41	10
11	18447	2013-07-03T19:21:44	11
12	18447	2013-07-03T20:27:05	12
13	18447	2013-07-03T20:42:58	13
14	18447	2013-07-03T20:53:11	14
15	18447	2013-07-03T21:54:49	15

페이지당 결과 수: 50 1 – 50 (전체 5975행)

개인 기록 프로젝트 기록 저장된 쿼리

Chapter.

유저의 구매순서를 구해보기 (feat. 첫구매)

Q. 가장 많이 구매한 유저의 첫번째로 구매한 상품의 카테고리는?

Q. 정류소별로 가장 마지막에 빌려진 자전거의 대여시각을 구하라.

코호트 분석

https://www.datarian.io/blog/cohort-analysis

Chapter.

사용자 지정 함수 UDF(User Define Function)

UDF?

사용자 지정 함수 UDF(**U**ser **D**efine **F**unction).
말그래도 사용자의 입맛대로 커스텀해서 만든 함수를 지칭.

복잡한 연산 등을 해야하는 경우 기본 함수로만으로는 분석에 한계에 있는 경우가 종종 발생.
해서, 사용자가 원하는 함수를 직접 등록해서 범용적으로 사용하기 위한 목적.

—-

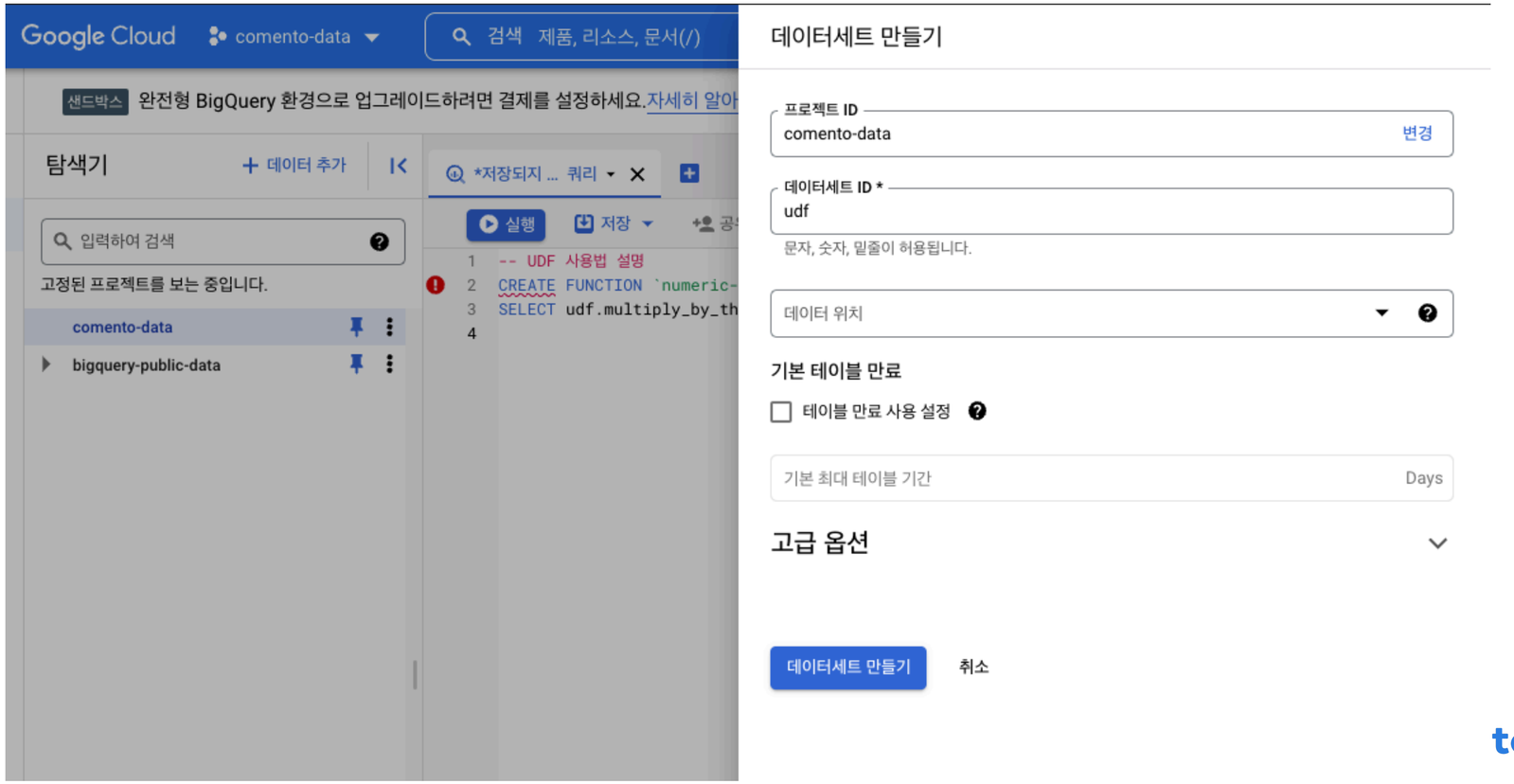
e.g. 두 개의 위경도를 통해 지리적 거리를 구하는 방법. (Harversine Formula)

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

데이터셋 만들기

프로젝트ID 설정에서 데이터세트 만들기 클릭.

“udf"라고 데이터세트ID 명을 입력후 생성하기.



UDF 직접 등록해보기

참고 : https://stackoverflow.com/questions/42068651/haversine-distance-in-bigquery

```
CREATE FUNCTION udf.haversine(Lat1 FLOAT64, Lng1 FLOAT64, Lat2 FLOAT64, Lng2 FLOAT64) AS (
    2 * 6335
        * sqrt(
            pow(sin((radians(Lat2) - radians(Lat1)) / 2), 2)
            + cos(radians(Lat1))
            * cos(radians(Lat2))
            * pow(sin((radians(Lng2) - radians(Lng1)) / 2), 2)
        )
)

CREATE FUNCTION udf.RADIANS(x FLOAT64) AS (
  ACOS(-1) * x / 180
);

CREATE FUNCTION udf.RADIANS_TO_KM(x FLOAT64) AS (
  111.045 * 180 * x / ACOS(-1)
);

CREATE FUNCTION udf.HAVERSINE(lat1 FLOAT64, long1 FLOAT64,
                              lat2 FLOAT64, long2 FLOAT64) AS (
  udf.RADIANS_TO_KM(
    ACOS(COS(udf.RADIANS(lat1)) * COS(udf.RADIANS(lat2)) *
         COS(udf.RADIANS(long1) - udf.RADIANS(long2)) +
         SIN(udf.RADIANS(lat1)) * SIN(udf.RADIANS(lat2))))
);
```

```
33
34  -- 대여지 : 37.548561 127.045006
35  -- 반납지 : 37.524837 126.934906
36  SELECT
37    udf.HAVERSINE(37.548561, 127.045006, 37.524837, 126.934906) AS distance_in_km
38
39
40
```

Alt+F1을

쿼리 결과

결과 저장

작업 정보 | 결과 | JSON | 실행 세부정보

행	distance_in_km
1	10.0463722009661…

Chapter.

이번 6주 과정으로 알아갈 것.
(매 시간마다 Wrap-up)

1주차 2주차 3주차 4주차 5주차 6주차

22.09.25(일) 20:00~23:00

1주차
데이터는 어떻게 저장되고 어떻게 추출해야 하나요?

01 오리엔테이션
- 멘토 및 클래스메이트 자기소개
- 클래스 대상 및 목표 안내
- 전체 커리큘럼 안내

02 현직자 이야기 : 비전공자가 데이터 분석가로 성장하기까지

03 SQL과 DBMS의 이해
- SQL, DB, TABLE, DBMS 용어의 이해

04 데이터 파이프라인의 이해
- 데이터는 어떻게 적재되어지는가

05 FROM WHERE SELECT 활용하기
- 특정 테이블에서 원하는 데이터 추출하기

06 SQL 쉽게 읽는 템플릿
- "바구니에서 빨간 연필을 꺼낸다."

07 실시간 Q&A

과제 FROM WHERE SELECT 활용하기
- 구글 빅쿼리 계정 발급
- 공공데이터(자전거대여BikeShare) 데이터 추출 연습 진행
- 프로그래머스 SQL 연습 (SELECT 문제)

22.10.02(일) 20:00~23:00

2주차
데이터를 "집계"해서 봅시다

01 1주차 세션 회고 및 과제 피드백

02 GROUP BY
- 집계 함수 정의, 그룹 별 집계 방법

03 HAVING
- 그룹 별로 조건을 걸어 원하는 데이터 추출하기

04 ORDER BY
- 오름차순 또는 내림차순으로 정렬하기

05 현직자 이야기 : 데이터 요청 잘하는 방법

06 실시간 Q&A

과제 GROUP BY, HAVING, ORDER BY 활용하기
- 공공데이터 (자전거대여BikeShare) 데이터 추출 연습 진행

22.10.23(일) 20:00~23:00

3주차
데이터들의 "조합"으로 뽑아봅시다

01 2주차 세션 회고 및 과제 피드백

02 JOIN
- JOIN의 종류, 결합키 찾기
- ERD 통한 테이블 관계 확인

03 SUB QUERY
- SUB QUERY의 종류(WITH절), 사용시 유의사항

04 현직자 이야기 : SQL을 잘 읽고 쓰는 법 (코드컨벤션 설명)

05 VSCODE 설치 및 소개

06 실시간 Q&A

과제 JOIN, SUB QUERY 활용하기
- 공공데이터(자전거대여BikeShare) 데이터 추출 연습 진행

1주차 2주차 3주차 **4주차** 5주차 6주차

22.10.30(일) 20:00~23:00

4주차
고객의 첫 주문이 알고 싶어요!

오늘은 여기까지 했습니다!

01 3주차 세션 회고 및 과제 피드백

02 조건문(CASE), UNION, EXCEPT

03 현업에서 자주 사용하는 함수 설명
- 내재함수 (TYPE, DESCRIBE, EXPLAIN)
- 날짜함수 (날짜 변환(e.g. 요일추출), 날짜 연산)
- 문자함수 (숫자를 문자로)
- 수학함수 (통계, 제곱)

04 고급함수 맛보기
- WINDOW 분석함수 (리텐션, 코호트)
- 정규표현식(휴대폰번호 추출)
- UDF(사용자 정의 함수)

05 실시간 Q&A

과제 함수 & WINODW 실습
- 유저의 첫 구매를 구하고, 주차별 리텐션율을 구해보기

4주차 강의를 모두 들으시느라 고생하셨습니다!

궁금한 내용은 언제든 문의주세요. :-)