

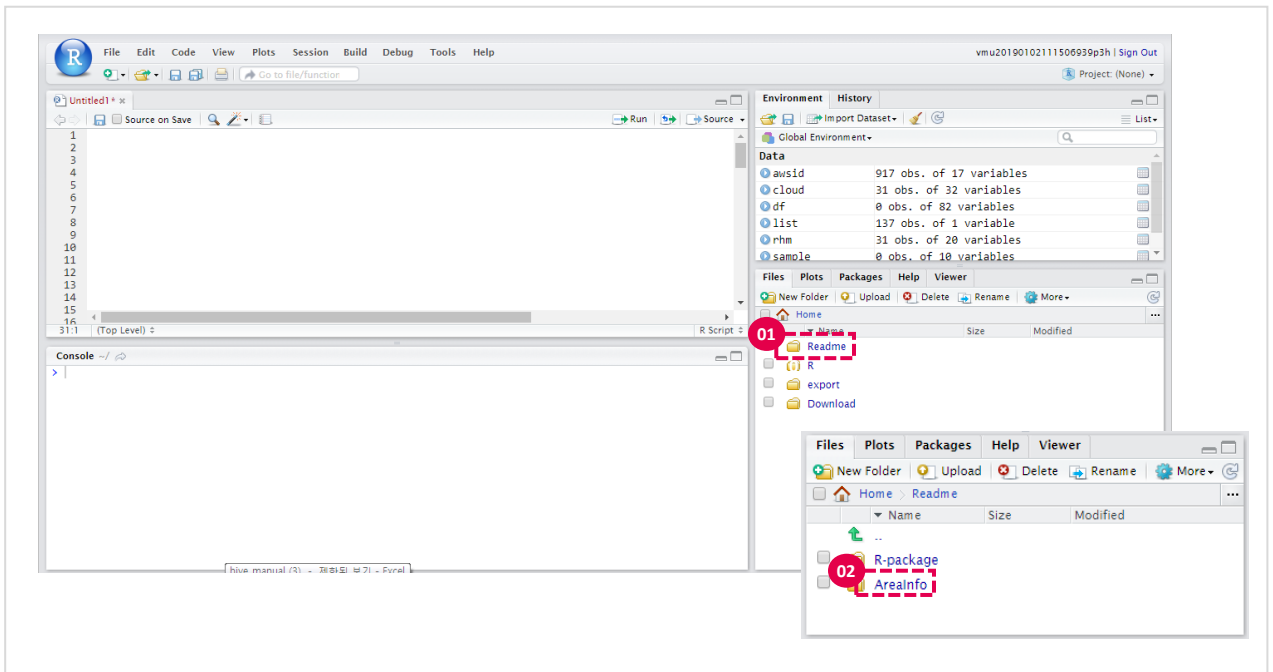
# HIVE 따라하기

하이프(HIVE)란, 하둡 파일 데이터를 HiveQL이라는 **SQL과 유사한 언어**를 사용하여 요약 및 분석을 수행할 수 있도록 도와주는 데이터 웨어하우스 시스템입니다.

하이브를 통해 데이터 테이블 추출 및 조인에 대해 알아보겠습니다.

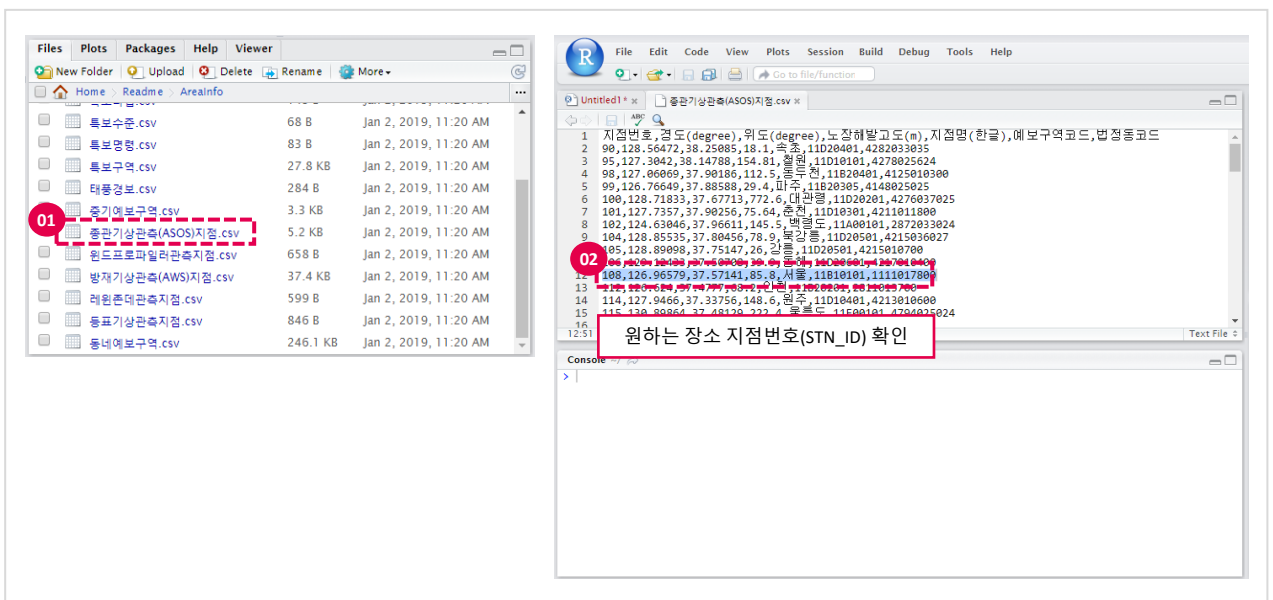
## STEP 1

### 관측지점 위치 확인



## STEP 2

### 특정장소 지점번호 확인 (ex : ASOS 기준)



ASOS 데이터에서 서울 관측지점(108)의 2020년 1월 한달 동안 기온, 습도, 구름 3가지 데이터 테이블을 추출하겠습니다.

### STEP 3 전체 테이블 리스트 확인 및 추출 테이블명 확인

The screenshot shows the R Studio environment. The main editor window displays an R script with the following code:

```

1
2 # 하이브내 전체 테이블 리스트 확인
3 list<-dbGetQuery(conn, "show tables")
4
5 View(list) # 전체 테이블 리스트 보기
6
7

```

A red dashed box highlights lines 3 and 5. A red circle with the number '01' is next to line 3. The status bar at the bottom indicates '(Top Level)' and 'R Script'.

To the right, the 'Environment' pane shows a list of tables from the database:

tab_name
1 aws_hr_hm
2 aws_hr_rm
3 aws_hr_ta
4 aws_hr_wd
5 db_aws_cloud_dd
6 db_aws_cloud_mnh
7 db_aws_cloud_tdom
8 db_aws_cloud_tim
9 db_aws_dd_old
10 db_aws_hm_tim_old
11 db_aws_icsr_ss_dd
12 db_aws_icsr_ss_mnh
13 db_aws_icsr_ss_tdom
14 db_aws_lwt_tg_dd
15 db_aws_nmyr_dd
16 db_aws_nmyr_mnh
17 db_aws_nmyr_sesn
18 db_aws_prsr_dd
19 db_aws_rhm_dd
20 db_aws_rhm_mnh

Below the table list, it says 'Showing 1 to 20 of 138 entries'.

A pink callout box points to the script with the text:

**추출 테이블 명**  
 ASOS 일별 기온 테이블 : db\_sfc\_ta\_dd  
 ASOS 일별 습도 테이블 : db\_sfc\_rhm\_dd  
 ASOS 일별 구름 테이블 : db\_sfc\_cloud\_dd

A pink box at the bottom contains the text:

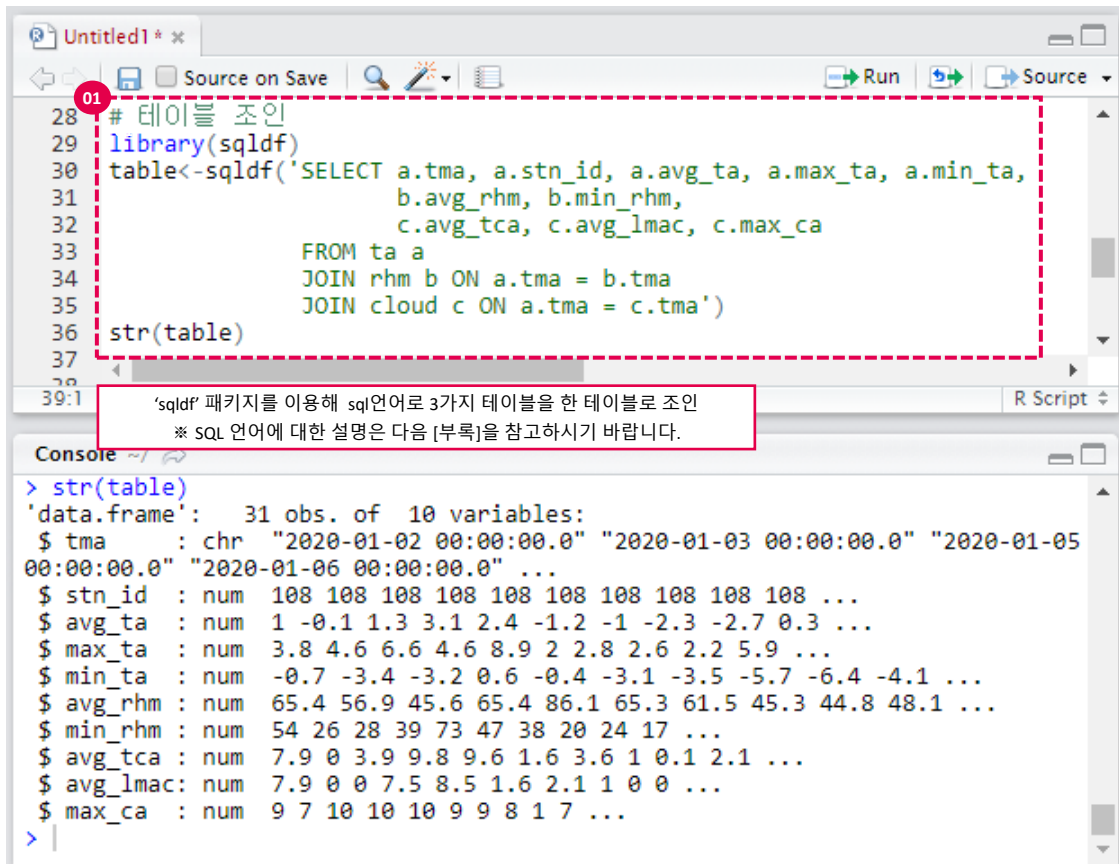
※ 테이블명에 대한 설명은 날씨마루 > 분석환경 > 데이터 > 기상데이터의 HIVE manual 참고하시기 바랍니다.

## STEP 4 데이터 테이블 추출

```
1 01 # ASOS 기온/습도/구름 테이블 추출
14 # 추출: seoul(stn_id:108), date(2020-01-01~2020-01-31)
15 #
16 ta<-dbGetQuery(conn, "SELECT * FROM db_sfc_ta_dd
17                        WHERE db_sfc_ta_dd.stn_id==108 and db_sfc_ta_dd.tma LIKE '2020-01%'")
18 rhm<-dbGetQuery(conn, "SELECT * FROM db_sfc_rhm_dd
19                       WHERE db_sfc_rhm_dd.stn_id==108 and db_sfc_rhm_dd.tma LIKE '2020-01%'")
20 cloud<-dbGetQuery(conn, "SELECT * FROM db_sfc_cloud_dd
21                         WHERE db_sfc_cloud_dd.stn_id==108 and db_sfc_cloud_dd.tma LIKE '2020-01%'")
22
23 02 # 칼럼명 단순화
24 colnames(ta)=gsub("db_sfc_ta_dd.", "", colnames(ta), ignore.case = T)
25 colnames(rhm)=gsub("db_sfc_rhm_dd.", "", colnames(rhm), ignore.case = T)
26 colnames(cloud)=gsub("db_sfc_cloud_dd.", "", colnames(cloud), ignore.case = T)
27
28
29:15 (Top Level) ↕ R Script ↕
```

STEP4에서 추출한 기온, 습도, 구름 3가지 데이터 테이블에서 날짜를 기준으로 필요한 변수를 뽑아 한테이블로 조인하겠습니다.

#### STEP 5 테이블 조인



The screenshot shows an R Studio window with a script editor and a console. The script editor contains R code for joining three tables using the 'sqldf' package. A red dashed box highlights the code from line 28 to 36. A red circle with the number '01' is next to line 28. A red box with Korean text is positioned below the script editor, pointing to the 'sqldf' function. The console shows the output of the 'str(table)' command, displaying the structure of the resulting data frame with 31 observations and 10 variables.

```
28 # 테이블 조인
29 library(sqldf)
30 table<-sqldf('SELECT a.tma, a.stn_id, a.avg_ta, a.max_ta, a.min_ta,
31                b.avg_rhm, b.min_rhm,
32                c.avg_tca, c.avg_lmac, c.max_ca
33                FROM ta a
34                JOIN rhm b ON a.tma = b.tma
35                JOIN cloud c ON a.tma = c.tma')
36 str(table)
```

'sqldf' 패키지를 이용해 sql언어로 3가지 테이블을 한 테이블로 조인  
※ SQL 언어에 대한 설명은 다음 [부록]을 참고하시기 바랍니다.

```
> str(table)
'data.frame':  31 obs. of  10 variables:
 $ tma      : chr  "2020-01-02 00:00:00.0" "2020-01-03 00:00:00.0" "2020-01-05
00:00:00.0" "2020-01-06 00:00:00.0" ...
 $ stn_id   : num  108 108 108 108 108 108 108 108 108 ...
 $ avg_ta   : num  1 -0.1 1.3 3.1 2.4 -1.2 -1 -2.3 -2.7 0.3 ...
 $ max_ta   : num  3.8 4.6 6.6 4.6 8.9 2 2.8 2.6 2.2 5.9 ...
 $ min_ta   : num  -0.7 -3.4 -3.2 0.6 -0.4 -3.1 -3.5 -5.7 -6.4 -4.1 ...
 $ avg_rhm  : num  65.4 56.9 45.6 65.4 86.1 65.3 61.5 45.3 44.8 48.1 ...
 $ min_rhm  : num  54 26 28 39 73 47 38 20 24 17 ...
 $ avg_tca  : num  7.9 0 3.9 9.8 9.6 1.6 3.6 1 0.1 2.1 ...
 $ avg_lmac : num  7.9 0 0 7.5 8.5 1.6 2.1 1 0 0 ...
 $ max_ca   : num  9 7 10 10 10 9 9 8 1 7 ...
```

# [부록] SQL 언어 이해하기 I

빅데이터 활용 측면에서 데이터베이스 언어를 이해하고 필요 부분의 활용능력을 습득하는 것은 분석자에게 있어 굉장히 유용합니다.

## SQL(Structured Query Language, 구조화 질의어)

**(정의)** 관계형 데이터베이스 관리 시스템(RDBMS)의 데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어

**(필요성)** RDBMS에서 자료의 검색과 관리, 데이터베이스 스키마 생성과 수정, 데이터베이스 객체 접근 조정 관리를 위해 고안되었으며, 많은 수의 데이터베이스 관련 프로그램의 표준 언어로 채택

[ 출처:

위키백과 ]

## SQL 기본 문법

select 선택변수	# 추출 변수선택
from 데이터가 저장된 테이블	# 변수가 포함된 테이블 지정
[where 데이터 추출 조건]	# 추출 조건부여
[group by 기준변수]	# 데이터 그룹화
[Having 기준변수]	# 그룹화된 후 필터링
[order by 기준변수]	# 데이터 정렬
* [ ] 부분은 생략 가능	

## SQL 응용 문법

### ① select

모든 변수를 선택하고 싶으면 \*, 중복제거는 distinct

새로운 변수 생성은 select f(x) as y # f(x)를 새로운 변수 y로 정의

요약 통계량 계산시 count, avg, sum (groub by 사용시, 그룹안 통계량 계산)

### ② where

and, or, between, in, like, not 등의 명령어 사용하면 조건 추출에 용이

예시) where yymmdd between 20180101 and 20180630 # 18년 1~6월(상반기) 추출

x in (select x from data) # 다른 데이터 x조건에서 추출

x not in ('a', 'b', 'c') # a, b, c가 없는 변수 x 추출

substr (yymmdd,5,2)=01 # 1월 데이터만 추출

\* substr(문자열, 시작점, 길이): 일정 문자열에서 선택한 길이만큼 자름

# [부록] SQL 언어 이해하기 II

## SQL 연산자

테이블에서 요구 조건이 복잡해질 수록 WHERE 절에 조건식에서 다양한 연산자들을 사용해서 조건을 완성할 수 있습니다.

WHERE 절에 사용되는 연산자는 3가지 종류가 있습니다.

- 비교 연산자 (부정 비교 연산자 포함)
- SQL 연산자 (부정 SQL 연산자 포함)
- 논리 연산자

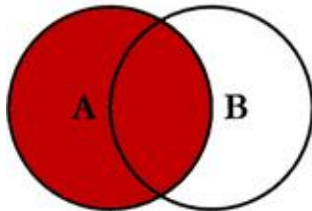
구분	연산자	연산자의 의미
비교 연산자	=	같다.
	>	보다 크다.
	>=	보다 크거나 같다.
	<	보다 작다.
	<=	보다 작거나 같다.
SQL 연산자	BETWEEN a AND b	a와 b의 값 사이에 있으면 된다.(a와 b 값이 포함됨)
	IN (list)	리스트에 있는 값 중에서 어느 하나라도 일치하면 된다.
	LIKE '비교문자열'	비교문자열과 형태가 일치하면 된다.(%, _ 사용)
	IS NULL	NULL 값인 경우
논리 연산자	AND	앞에 있는 조건과 뒤에 오는 조건이 참(TRUE)이 되면 결과도 참(TRUE)이 된다. 즉, 앞의 조건과 뒤의 조건을 동시에 만족해야 한다.
	OR	앞의 조건이 참(TRUE)이거나 뒤의 조건이 참(TRUE)이 되어야 결과도 참(TRUE)이 된다. 즉, 앞뒤의 조건 중 하나만 참(TRUE)이면 된다.
	NOT	뒤에 오는 조건에 반대되는 결과를 되돌려 준다.
부정 비교 연산자	!=	같지 않다.
	^=	같지 않다.
	◇	같지 않다.(ISO 표준, 모든 운영체제에서 사용 가능)
	NOT 컬럼명 =	~와 같지 않다.
	NOT 컬럼명 >	~보다 크지 않다.
부정 SQL 연산자	NOT BETWEEN a AND b	a와 b의 값 사이에 있지 않다. (a, b 값을 포함하지 않는다)
	NOT IN (list)	list 값과 일치하지 않는다.
	IS NOT NULL	NULL 값을 갖지 않는다.

# [부록] SQL 언어 이해하기 III

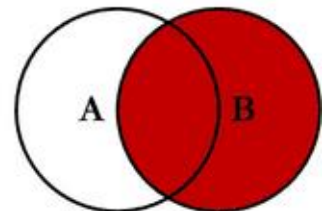
## SQL 연산자

JOIN은 2개의 테이블에서 각각의 공통값을 이용함으로써 필드를 조합하는 수단으로, 한 데이터베이스 내의 여러 테이블의 레코드를 조합하여 **하나의 열로 표현**

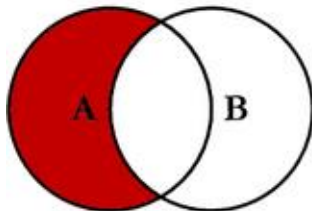
### SQL JOINS



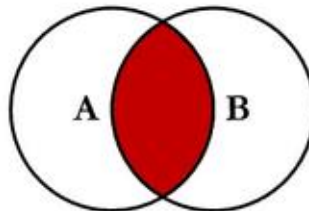
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



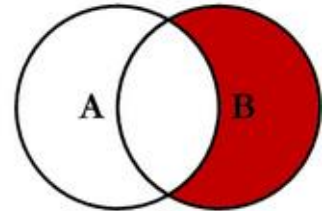
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



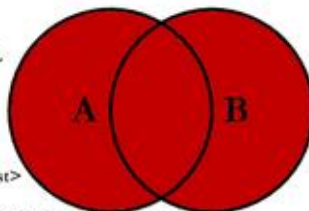
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



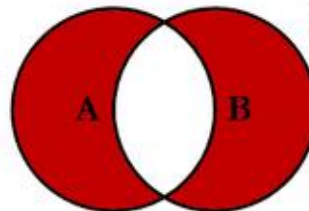
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



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



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Select A.변수, B.변수

from data1 as A (right/left/inner/full join) data2 as B  
on A.ID=B.ID

※ INNER JOIN 구문을 생략해 JOIN이라 써도 무방합니다.

※ R의 sqldf 패키지의 경우, right와 full join은 제공하지 않습니다