

# sql day2

## #비교 연산자

= 같다

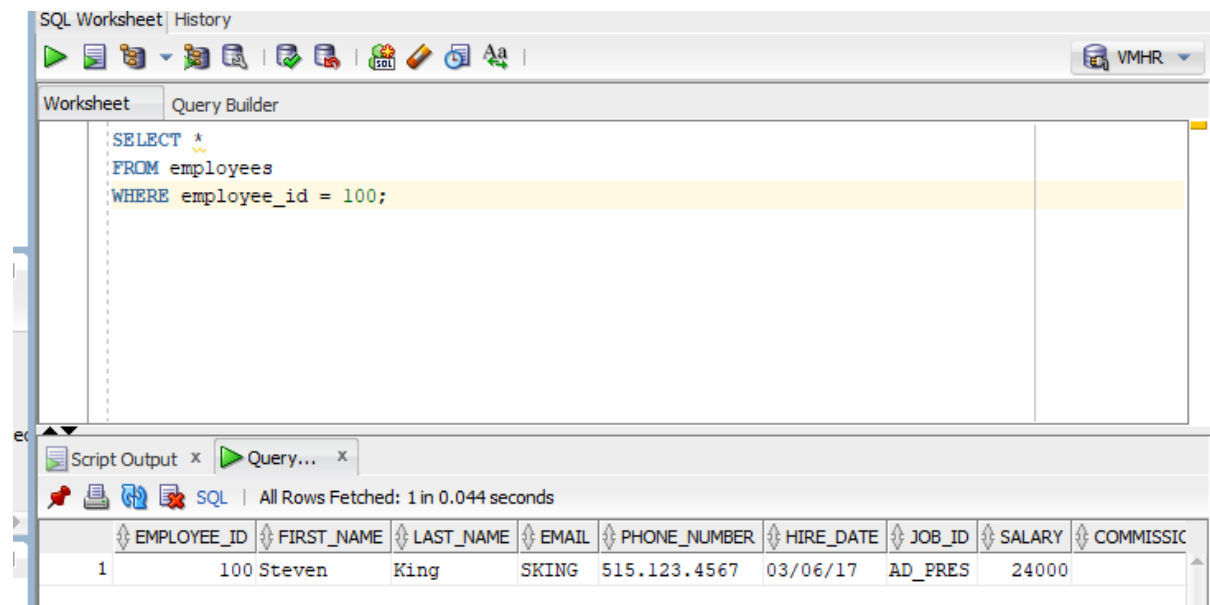
!= 다르다 (<>)

..> 크다

..< 작다

..>= 크거나 같다

..<= 작거나 같다



The screenshot shows a SQL Worksheet interface with a query editor and a results table. The query is:

```
SELECT *  
FROM employees  
WHERE employee_id = 100;
```

The results table shows one row of data for employee\_id 100.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSIC
1	100	Steven	King	SKING	515.123.4567	03/06/17	AD_PRES	24000

## #논리 연산자

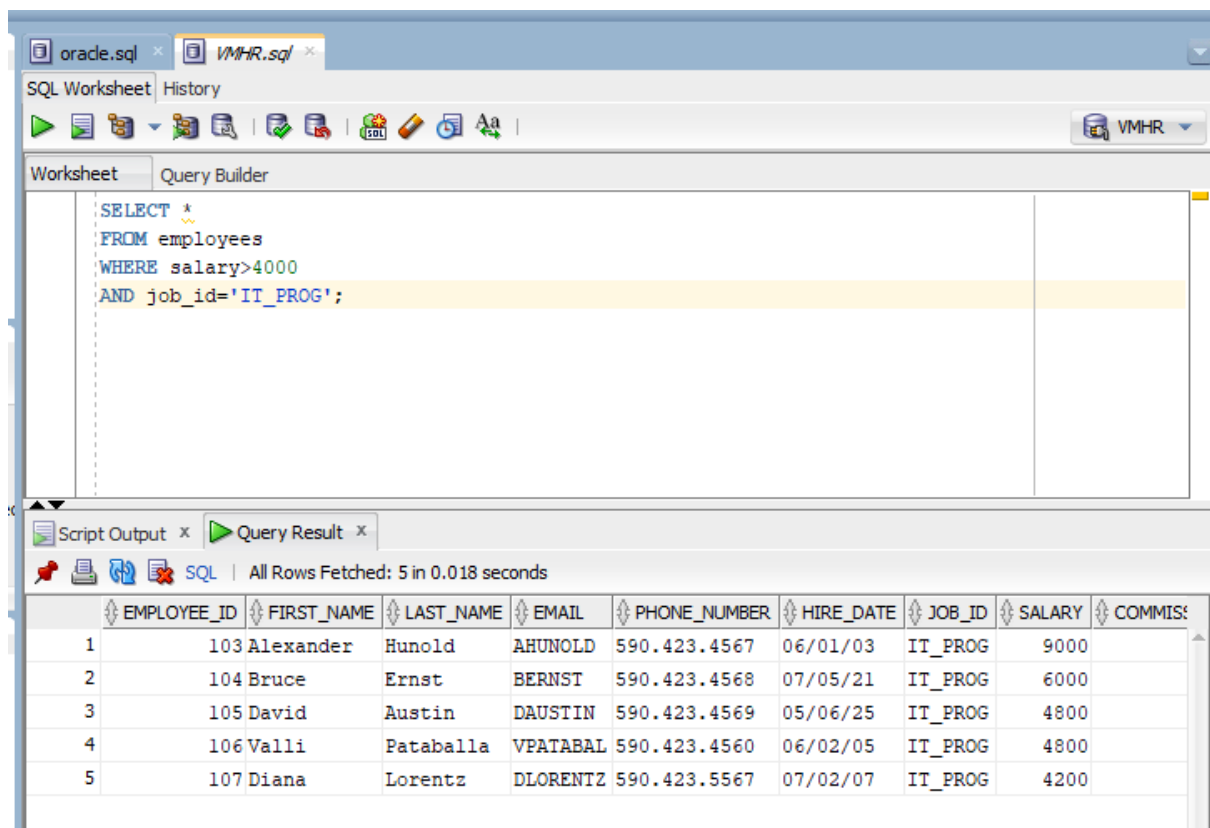
AND 모든 조건을 동시에 다 만족할 때만 true

OR 조건 중 하나만 만족해도 true

NOT 조건의 반대 결과를 반환

```
SELECT *  
FROM 테이블  
WHERE 조건1  
AND 조건2;
```

cf.“,” 동일하지만 // “선호한다.



The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the following SQL query:

```
SELECT *  
FROM employees  
WHERE salary>4000  
AND job_id='IT_PROG';
```

The 'Query Result' tab is also visible, showing the results of the query. The status bar indicates 'All Rows Fetched: 5 in 0.018 seconds'. The results are displayed in a table with the following columns: EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, and COMMISSION\_PCT.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
1	103 Alexander	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	
2	104 Bruce	Ernst	BERNST	590.423.4568	07/05/21	IT_PROG	6000	
3	105 David	Austin	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800	
4	106 Valli	Pataballa	VPATABAL	590.423.4560	06/02/05	IT_PROG	4800	
5	107 Diana	Lorentz	DLORENTZ	590.423.5567	07/02/07	IT_PROG	4200	

Oracle SQL Worksheet interface showing a query and its results.

**SQL Worksheet:**

```

SELECT *
FROM employees
WHERE salary > 4000
AND job_id = 'IT_PROG'
OR job_id = 'FI_ACCOUNT';

```

**Query Result:** All Rows Fetched: 10 in 0.007 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COM
1	103	Alexander	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	
2	104	Bruce	Ernst	BERNST	590.423.4568	07/05/21	IT_PROG	6000	
3	105	David	Austin	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800	
4	106	Valli	Pataballa	VPATABAL	590.423.4560	06/02/05	IT_PROG	4800	
5	107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/07	IT_PROG	4200	
6	109	Daniel	Faviet	DFAVIET	515.124.4169	02/08/16	FI_ACCOUNT	9000	
7	110	John	Chen	JCHEN	515.124.4269	05/09/28	FI_ACCOUNT	8200	

Oracle SQL Worksheet interface showing a query and its results.

**SQL Worksheet:**

```

SELECT *
FROM employees
WHERE employee_id <> 105;

```

**Query Result:** Fetched 50 rows in 0.01 seconds

	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	515.123.4567	03/06/17	AD_PRES	24000	(null)	(null)	90
2 AR	515.123.4568	05/09/21	AD_VP	17000	(null)	100	90
3 N	515.123.4569	01/01/13	AD_VP	17000	(null)	100	90
4 D	590.423.4567	06/01/03	IT_PROG	9000	(null)	102	60
5	590.423.4568	07/05/21	IT_PROG	6000	(null)	103	60
6 AL	590.423.4560	06/02/05	IT_PROG	4800	(null)	103	60
7 IZ	590.423.5567	07/02/07	IT_PROG	4200	(null)	103	60

SQL Worksheet History

Worksheet Query Builder

```
SELECT *
FROM employees
WHERE manager_id IS NULL;
```

Script Output x Explain Plan x Query Result x

SQL | All Rows Fetched: 1 in 0.006 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSIC
1	100	Steven	King	SKING	515.123.4567	03/06/17	AD_PRES	24000	

SQL Worksheet History

Worksheet Query Builder

```
SELECT *
FROM employees
WHERE manager_id IS NOT NULL;
```

Script Output x Explain Plan x Query Result x

SQL | Fetched 50 rows in 0.014 seconds

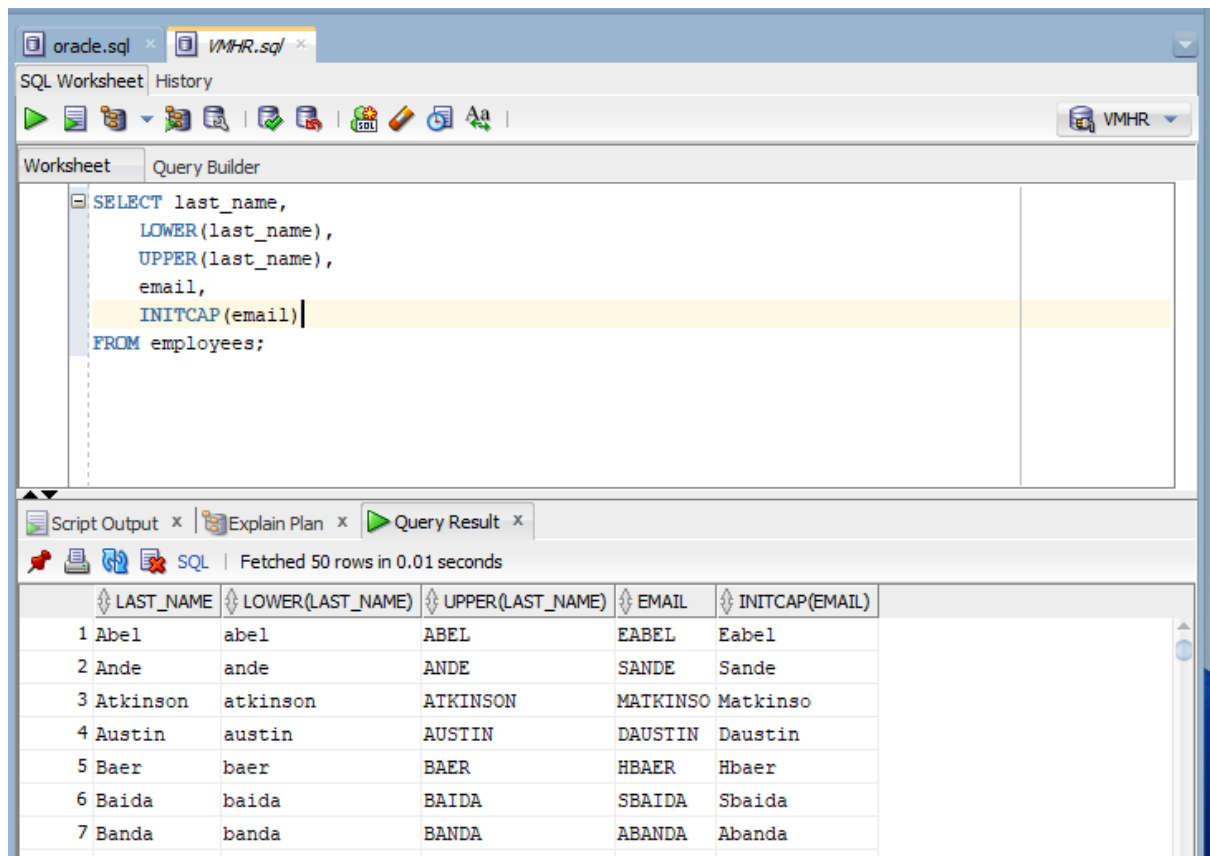
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	05/09/21	AD_VP	17000
2	102	Lex	De Haan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000
4	104	Bruce	Ernst	BERNST	590.423.4568	07/05/21	IT_PROG	6000
5	105	David	Austin	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800
6	106	Valli	Pataballa	VPATABAL	590.423.4560	06/02/05	IT_PROG	4800
7	107	Diana	Lorentz	DLORENTZ	590.423.5567	07/02/07	IT_PROG	4200

## #함수 사용

-단일행 함수: 특정 행에 적용 // 데이터 값을 하나씩 계산함

대문자/소문자/첫글자만 대문자(BOY/boy/Boy)

-그룹 함수: 그룹 전체 적용 // 여러개 값을 그룹으로 계산함



The screenshot shows an SQL Worksheet interface with a query editor and a results pane. The query editor contains the following SQL statement:

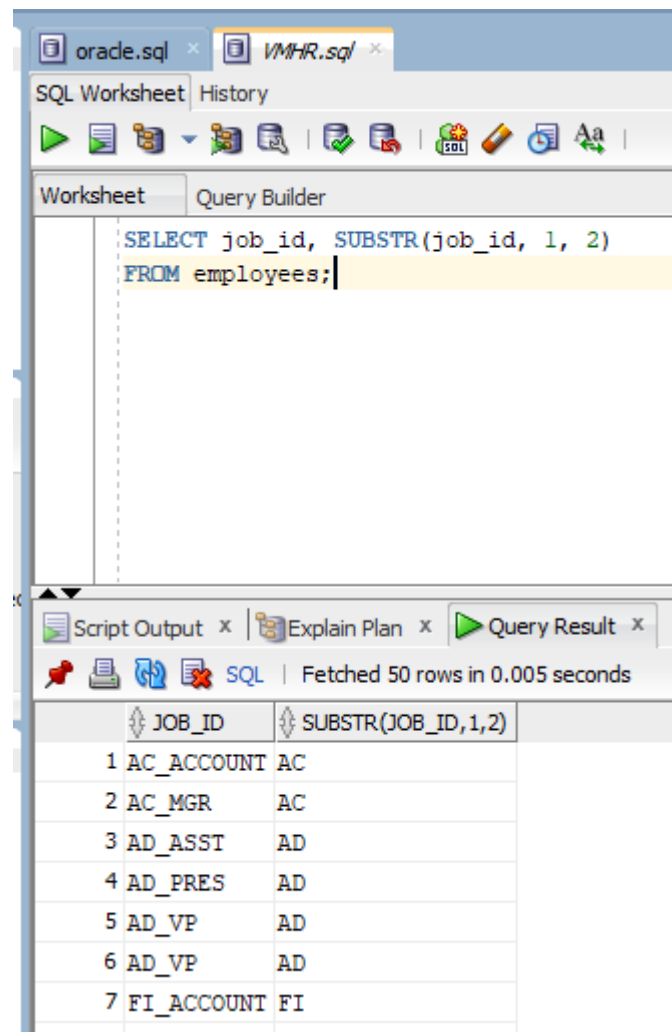
```
SELECT last_name,  
       LOWER(last_name),  
       UPPER(last_name),  
       email,  
       INITCAP(email)  
FROM employees;
```

The results pane displays the output of the query, showing 50 rows. The first 7 rows are visible in the table below:

	LAST_NAME	LOWER(LAST_NAME)	UPPER(LAST_NAME)	EMAIL	INITCAP(EMAIL)
1	Abel	abel	ABEL	EABEL	Eabel
2	Ande	ande	ANDE	SANDE	Sande
3	Atkinson	atkinson	ATKINSON	MATKINSO	Matkinso
4	Austin	austin	AUSTIN	DAUSTIN	Daustin
5	Baer	baer	BAER	HBAER	Hbaer
6	Baida	baida	BAIDA	SBAIDA	Sbaida
7	Banda	banda	BANDA	ABANDA	Abanda

## #글자 자르기 substr

SUBSTR('원본글자', 시작위치, 자를 개수)



#글자 바꾸기 ) 특정 문자를 찾아서 변경 replace

REPLACE('문자열','찾을 문자','바꿀 문자')

Worksheet		Query Builder	
		<pre>SELECT job_id, REPLACE(job_id, 'ACCOUNT', 'ACC') FROM employees;</pre>	
		<div> <div>Script Output x</div> <div>Explain Plan x</div> <div>Query Result x</div> </div> <div> <div>SQL</div> <div>Fetches 50 rows in 0.028 seconds</div> </div>	
JOB_ID	REPLACE(JOB_ID,'ACCOUNT','ACC')		
1 AC_ACCOUNT	AC_ACC		
2 AC_MGR	AC_MGR		
3 AD_ASST	AD_ASST		
4 AD_PRES	AD_PRES		
5 AD_VP	AD_VP		
6 AD_VP	AD_VP		

## #LPAD, RPAD - 특정 문자로 자리 채우기

왼쪽부터 채우기, 오른쪽부터 채우기

LPAD('문자열', '만들어질 자리수', '채울 문자')

SQL Worksheet History

Worksheet Query Builder

```
SELECT first_name, LPAD(first_name, 12, '*')
FROM employees;
```

Script Output x Explain Plan x Query Result x

SQL | Fetched 50 rows in 0.017 seconds

	FIRST_NAME	LPAD(FIRST_NAME,12,'*')
1	Ellen	*****Ellen
2	Sundar	*****Sundar
3	Mozhe	*****Mozhe
4	David	*****David
5	Hermann	*****Hermann
6	Shelli	*****Shelli
7	Amit	*****Amit
8	Elizabeth	***Elizabeth



Oracle SQL Developer interface showing a query and its results.

**SQL Worksheet:**

```
SELECT first_name, LPAD(first_name, 12, ' ')
FROM employees;
```

**Query Result:** Fetches 50 rows in 0.008 seconds.

	FIRST_NAME	LPAD(FIRST_NAME,12,"")
1	Ellen	Ellen
2	Sundar	Sundar
3	Mozhe	Mozhe
4	David	David
5	Hermann	Hermann
6	Shelli	Shelli
7	Amit	Amit
8	Elizabeth	Elizabeth

The screenshot shows an SQL Worksheet interface with two tabs: 'orade.sql' and 'VMHR.sql'. The 'VMHR.sql' tab is active, displaying the following SQL query:

```
SELECT first_name, RPAD(first_name, 12, '*')
FROM employees;
```

Below the query editor, the 'Query Result' tab is selected, showing the results of the query. The results are displayed in a table with two columns: 'FIRST\_NAME' and 'RPAD(FIRST\_NAME, 12, '\*')'. The table contains 8 rows of data, with the first 7 rows visible in the screenshot.

	FIRST_NAME	RPAD(FIRST_NAME, 12, '*')
1	Ellen	Ellen*****
2	Sundar	Sundar*****
3	Mozhe	Mozhe*****
4	David	David*****
5	Hermann	Hermann*****
6	Shelli	Shelli*****
7	Amit	Amit*****
8	Elizabeth	Elizabeth****

## #LTRIM / RTRIM 삭제하기

LTRIM ('문자열'열이름, '삭제할문자')

Oracle SQL Developer interface showing a query execution.

**SQL Worksheet:**

```
SELECT job_id, LTRIM(job_id, 'F')
FROM employees;
```

**Query Result:** Fetches 50 rows in 0.006 seconds.

	JOB_ID	LTRIM(JOB_ID,'F')
1	AC_ACCOUNT	AC_ACCOUNT
2	AC_MGR	AC_MGR
3	AD_ASST	AD_ASST
4	AD_PRES	AD_PRES
5	AD_VP	AD_VP
6	AD_VP	AD_VP
7	FI_ACCOUNT	I_ACCOUNT
8	FT_ACCOUNT	T_ACCOUNT

The screenshot shows an SQL Worksheet with two tabs: 'orade.sql' and 'VMHR.sql'. The 'VMHR.sql' tab is active, displaying the following SQL query:

```
SELECT job_id, LTRIM(job_id, 'F'), RTRIM(job_id, 'T')
FROM employees;
```

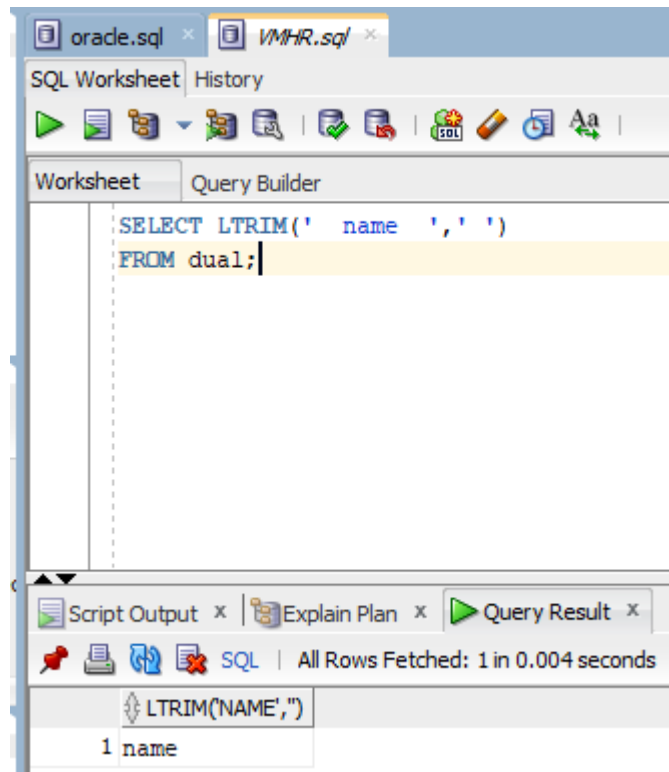
Below the query, the 'Query Result' tab is active, showing the results of the query. The results are displayed in a table with 4 columns: 'JOB\_ID', 'LTRIM(JOB\_ID,'F')', and 'RTRIM(JOB\_ID,'T')'. The table contains 7 rows of data.

	JOB_ID	LTRIM(JOB_ID,'F')	RTRIM(JOB_ID,'T')
1	AC_ACCOUNT	AC_ACCOUNT	AC_ACCOUN
2	AC_MGR	AC_MGR	AC_MGR
3	AD_ASST	AD_ASST	AD_ASS
4	AD_PRES	AD_PRES	AD_PRES
5	AD_VP	AD_VP	AD_VP
6	AD_VP	AD_VP	AD_VP
7	FI_ACCOUNT	I_ACCOUNT	FI_ACCOUN

★테이블을 쓸 건 아니지만 문법상 명시해 주어야 할 때

FROM dual; 사용

-dual 테이블은 dummy 테이블, 특정 테이블을 사용하지 않고 문법적으로 오류를 회피하고자 할 때 사용하는 일종의 가상의 테이블로 생각하자.

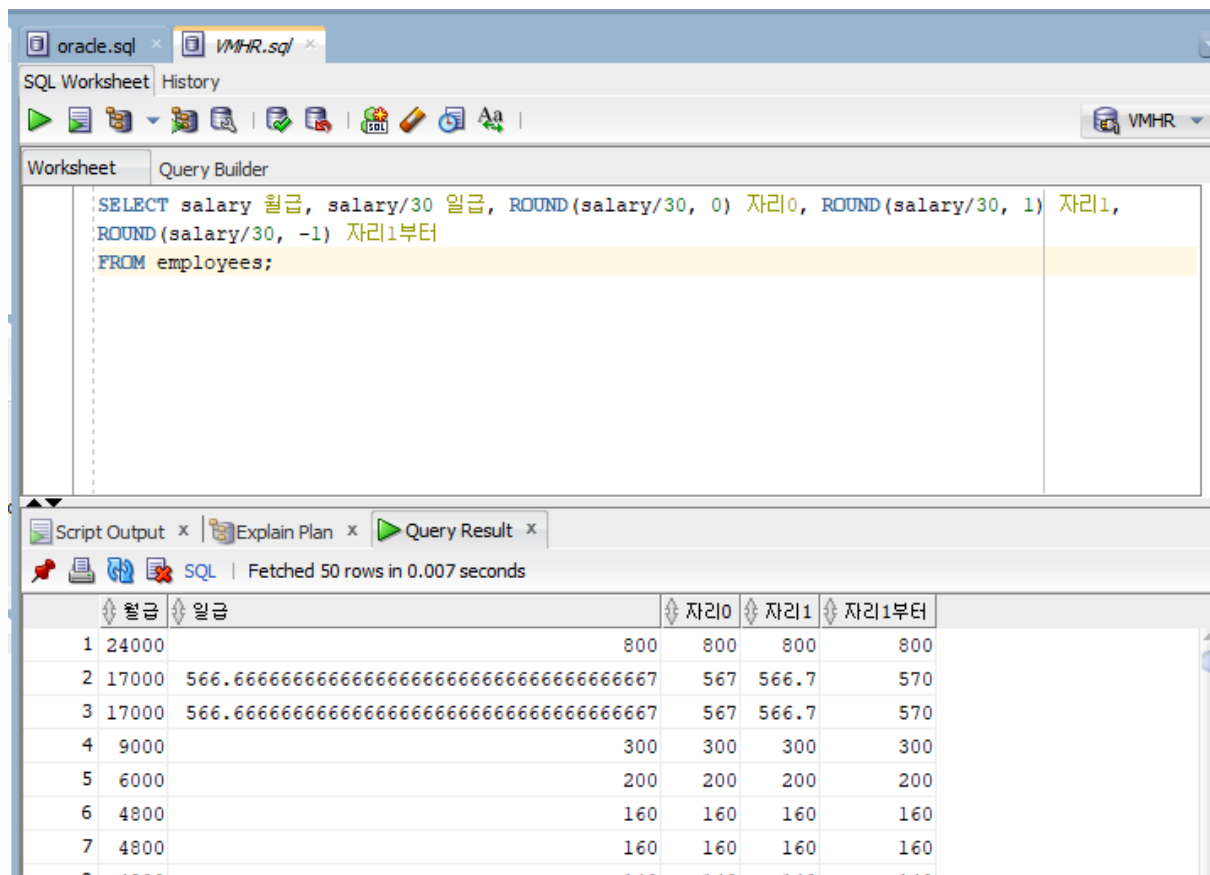


## #문자 관련 함수들

-숫자관련 함수들

round-반올림

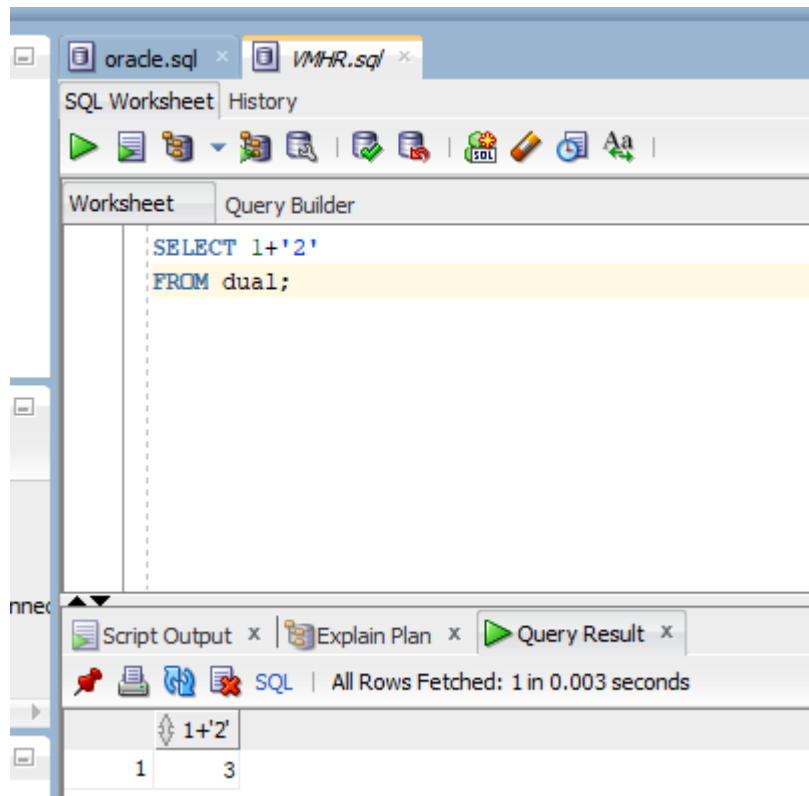
ROUND('열이름', 반올림할 위치)



## #trunc 버림/절삭

trunc(열이름, 자리값)





●숫자 2의 경우는 작은 따옴표가 붙어 있어 숫자가 아닌 문자이다. 하지만 예외가 발생하지 않고 원하는 방향으로 계산이 이루어졌다. 시스템이 자동으로 알아서 숫자로 변환하여 계산하였기 때문이다.

●이렇듯 자동으로 형변환이 이루어 질 수는 있지만, 항상 사용자가 원하는 의도대로 변환이 자동으로 완벽하게 이루어지지 않는다. 따라서 자동 형변환이 잘 되더라도 **수동으로 의도적으로 명시해 주는 것이 바람직하다.**

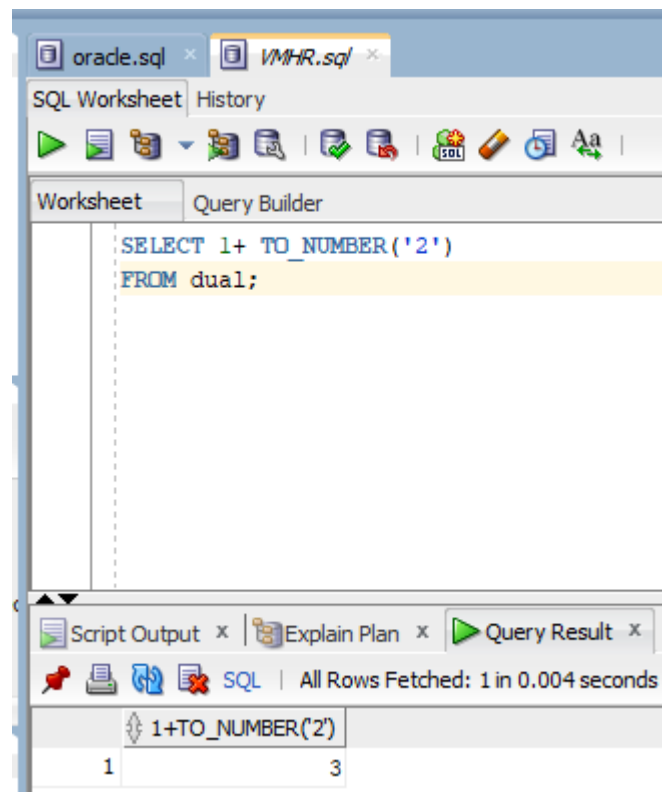
**-수동 형변환(명시적 형변환)**수동으로 데이터 형 변환

TO\_CHAR 문자로 형변환

TO\_NUMBER 숫자로 형변환

TO\_DATE 날짜로 형변환





## #VARCHAR

VARCHAR2(varchar) → NUMBER(integer)

NUMBER → VARCHAR2

SQL Worksheet: VMHR.sql

Worksheet: Query Builder

```
SELECT *
FROM employees
ORDER BY commission_pct;
```

Script Output | Explain Plan | Query Result

SQL | Fetched 50 rows in 0.012 seconds

IE	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGI
34	McEwen	AMCEWEN	011.44.1345.829268	04/08/01	SA_REP	9000	0.35	
35	Russell	JRUSSEL	011.44.1344.429268	04/10/01	SA_MAN	14000	0.4	
36	King	SKING	515.123.4567	03/06/17	AD_PRES	24000	(null)	(t
37	Kochhar	NKOCHHAR	515.123.4568	05/09/21	AD_VP	17000	(null)	
38	De Haan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000	(null)	
39	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	(null)	
40	Ernst	BERNST	590.423.4568	07/05/21	IT_PROG	6000	(null)	

Line 3 Column 25 | Insert | Modified | Windows: Cf

-계산 시 null 값이 문제가 된다.

SQL Worksheet History

Worksheet Query Builder

```
SELECT salary * commission_pct
FROM employees
ORDER BY commission_pct;
```

Script Output x Explain Plan x Query Result x

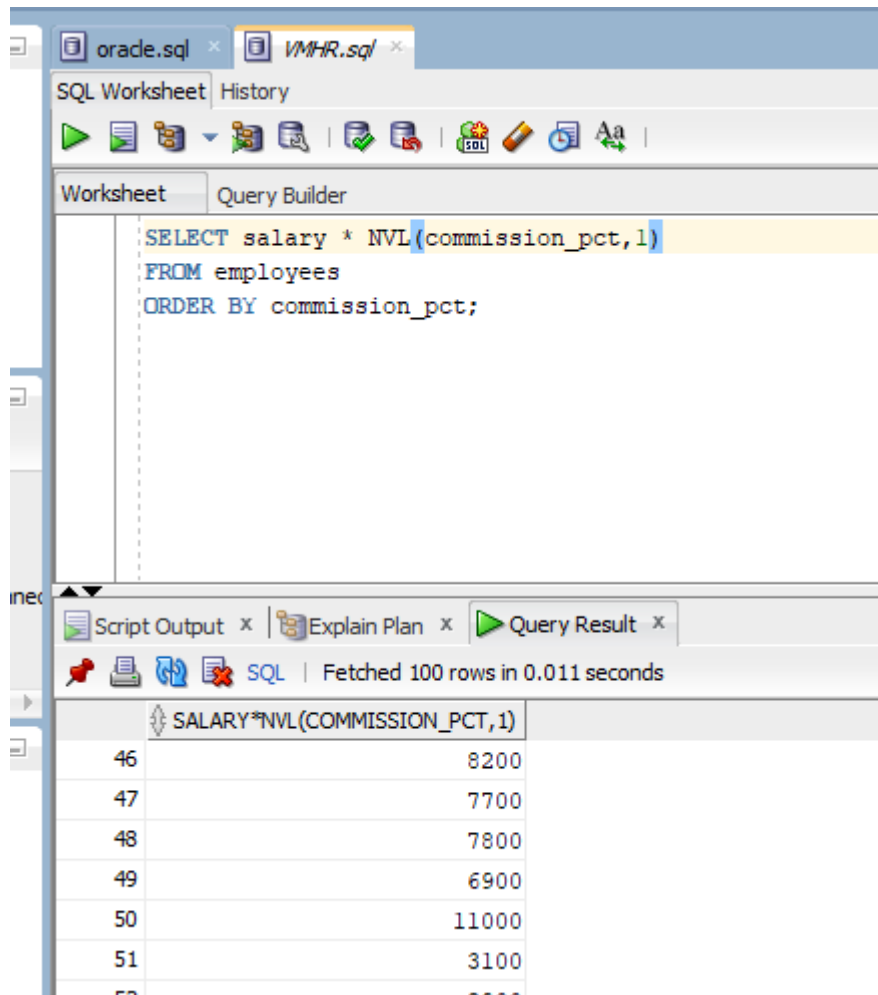
SQL | Fetched 100 rows in 0.01 seconds

	SALARY*COMMISSION_PCT
31	2250
32	3500
33	3325
34	3150
35	5600
36	(null)
37	(null)

-salary와 commission\_pct 값을 계산하려고 하면 문제가 발생 → 이유는 커미션을 지급받지 않는 직원이 있기 때문이다. (null값)

-null로 계산하면 결과는 null이 된다. (null을 처리하는 것이 중요)

-null값을 1로 치환하여 계산하면 전체 계산에 문제를 발생시키지 않게 된다. (곱해도 나뉘어도 본인이 나오므로...)



-null 오류가 없어짐

## #NVL 함수

null값을 특정값으로 치환하여 계산이 이루어지도록 처리한다.

## #DECODE ) 조건 처리하기

DECODE(열 이름, 조건값, 치환값, 기본값)

-치환값)조건을 만족할 경우

-기본값)조건을 만족하지 않을 경우

Oracle SQL Worksheet showing the following query:

```
SELECT department_id, employee_id, first_name, salary 원래급여,
DECODE(department_id, 60, salary*1.1, salary)
FROM employees;
```

Query Results (7 rows):

	DEPARTMENT_ID	EMPLOYEE_ID	FIRST_NAME	원래급여	DECODE(DEPARTMENT_ID,60,SALARY*1.1,SALARY)
1	90	100	Steven	24000	24000
2	90	101	Neena	17000	17000
3	90	102	Lex	17000	17000
4	60	103	Alexander	9000	9900
5	60	104	Bruce	6000	6600
6	60	105	David	4800	5280
7	60	106	Valli	4800	5280

```
SELECT department_id, employee_id, first_name, salary 원래급여,
DECODE(department_id, 60, salary*1.1, salary)
FROM employees;
```

고급화)

SQL Worksheet History

Worksheet Query Builder

```

SELECT department_id, employee_id, first_name, salary 원래급여,
DECODE(department_id, 60, salary*1.1, salary),
DECODE(department_id, 60, '급여인상', ' ')

FROM employees;

```

Script Output x Explain Plan x Query Result x

SQL | Fetched 50 rows in 0.006 seconds

	DEPARTMENT_ID	EMPLOYEE_ID	FIRST_NAME	원래급여	DECODE(DEPARTMENT_ID,60,SALARY*1.1,SALARY)	DECODE(C
1	90	100	Steven	24000	24000	
2	90	101	Neena	17000	17000	
3	90	102	Lex	17000	17000	
4	60	103	Alexander	9000	9900	급여인상
5	60	104	Bruce	6000	6600	급여인상
6	60	105	David	4800	5280	급여인상

#CASE ) 경우의 수가 여러가지인 경우, 복잡한 조건 처리

The screenshot shows a SQL query builder window with a 'Query Builder' tab. The query is as follows:

```
SELECT job_id, employee_id, first_name, salary,
CASE
    WHEN salary >= 9000 THEN '상급개발자'
    WHEN salary >= 5000 THEN '중급개발자'
    ELSE '하급개발자'
END AS 구분
FROM employees
WHERE job_id = 'IT_PROG';
```

Below the query, the 'Query Result' tab is active, displaying the results of the query. The status bar indicates 'All Rows Fetched: 5 in 0.01 seconds'.

	JOB_ID	EMPLOYEE_ID	FIRST_NAME	SALARY	구분
1	IT_PROG	103	Alexander	9000	상급개발자
2	IT_PROG	104	Bruce	6000	중급개발자
3	IT_PROG	105	David	4800	하급개발자
4	IT_PROG	106	Valli	4800	하급개발자
5	IT_PROG	107	Diana	4200	하급개발자

### #순위 매기기 3가지/numbering

- 1)RANK 공통 순위만큼 건너뛰어 순위 매기기 1,2,2,4
- 2)DENSE\_RANK 공통 순위를 건너 뛰지 않고 순위 매기기 1,2,2,3
- 3)ROW\_NUMBER 공통 순위 없이 출력 1,2,3,4

```
SELECT employee_id, first_name, salary,
RANK() OVER(ORDER BY salary DESC) rank순위,
DENSE_RANK() OVER(ORDER BY salary DESC) dense순위,
ROW_NUMBER() OVER(ORDER BY salary DESC) row순위
FROM employees;
```

The screenshot shows the Oracle SQL Developer interface. The top pane displays a SQL query in the 'SQL 워크시트(W) 내역' (SQL Worksheet) tab. The query is as follows:

```

SELECT employee_id, first_name, salary,
       RANK() OVER(ORDER BY salary DESC) rank순위,
       DENSE_RANK() OVER(ORDER BY salary DESC) dense순위,
       ROW_NUMBER() OVER(ORDER BY salary DESC) row순위
FROM employees;

```

The bottom pane shows the '질의 결과' (Query Results) tab, indicating that 50 rows were returned in 0.204 seconds. The results are displayed in a table with the following columns: EMPLOYEE\_ID, FIRST\_NAME, SALARY, RANK순위, DENSE순위, and ROW순위.

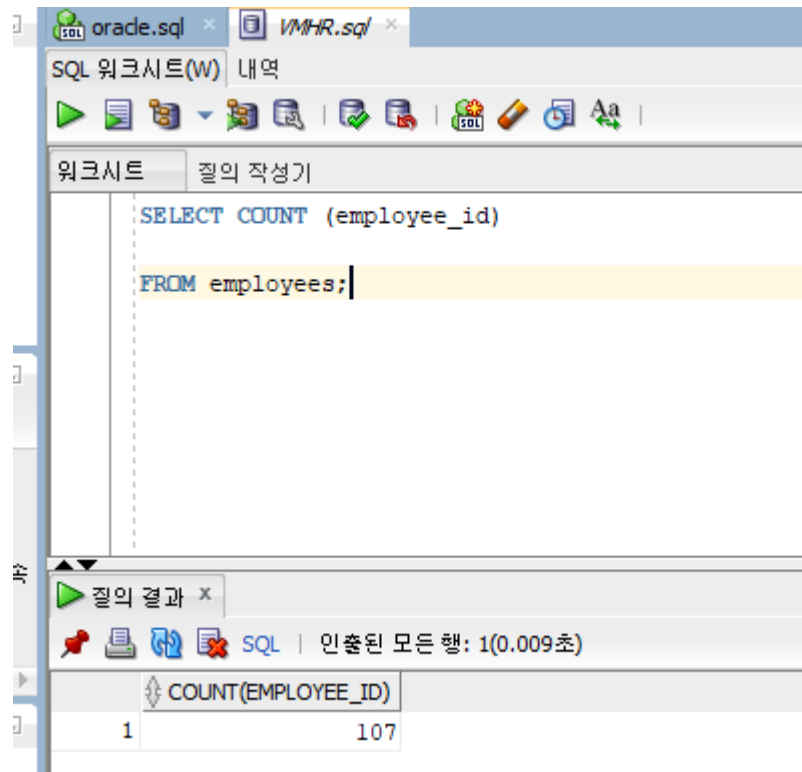
	EMPLOYEE_ID	FIRST_NAME	SALARY	RANK순위	DENSE순위	ROW순위
1	100	Steven	24000	1	1	1
2	101	Neena	17000	2	2	2
3	102	Lex	17000	2	2	3
4	145	John	14000	4	3	4
5	146	Karen	13500	5	4	5
6	201	Michael	13000	6	5	6
7	108	Nancy	12008	7	6	7
8	205	Shelley	12008	7	6	8

## #그룹 함수

여러 행에 함수가 적용되어 하나의 결과를 나타낸다.

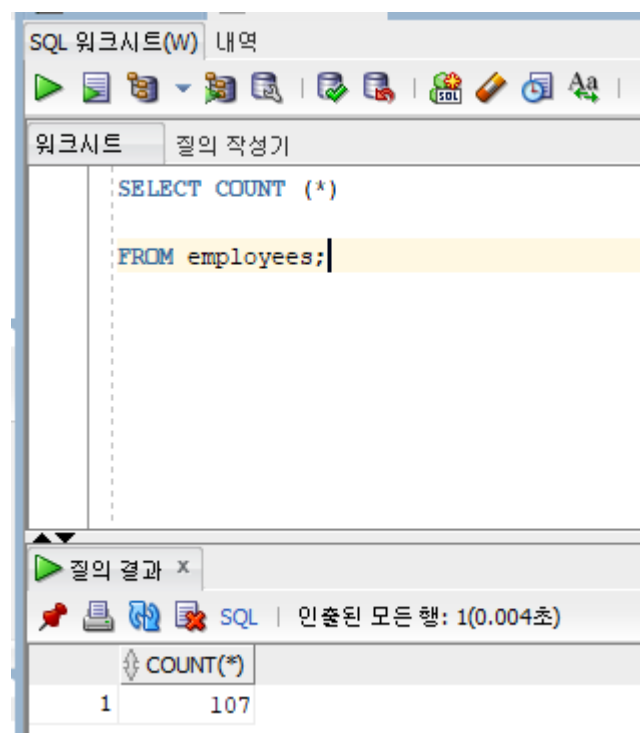
count 개수 ) null 값도 포함하여 계산한다.





count는 열이름을 생략하고 주로 \*로 사용한다.

어떤 것으로 세도 동일한 결과



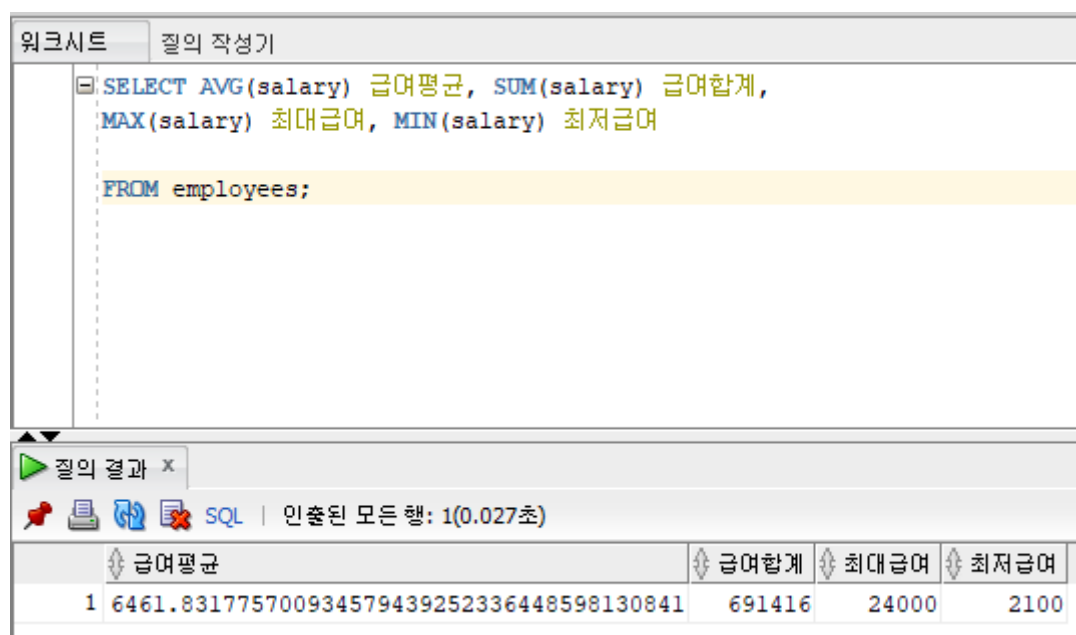
-count는 특성상 null도 계산하기 때문에 어떠한 열로도 동일한 결과값이 나오므로 \*를 주로 사용한다.

sum 합계 ) null 값을 제외하고 계산한다.

avg 평균 ) null 값을 제외하고 계산한다.

max 최대값 ) null 값을 제외하고 계산한다.

min 최소값 ) null 값을 제외하고 계산한다.



The screenshot shows a SQL IDE interface. The top pane, titled '워크시트' (Worksheet), contains the following SQL query:

```
SELECT AVG(salary) 급여평균, SUM(salary) 급여합계,  
MAX(salary) 최대급여, MIN(salary) 최저급여  
FROM employees;
```

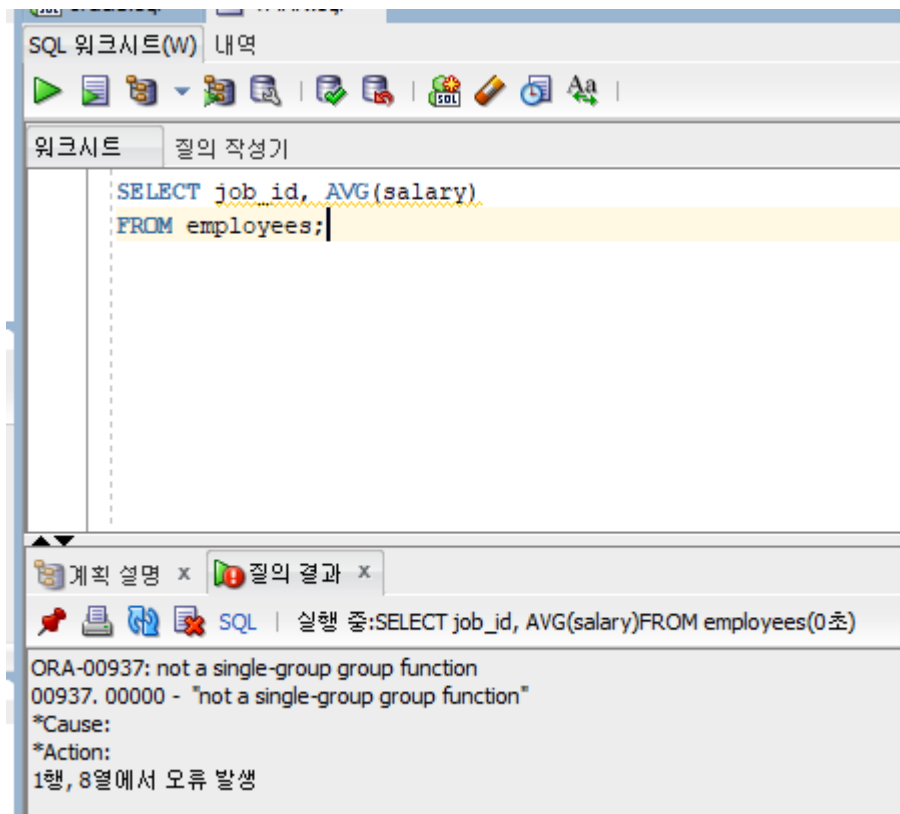
The bottom pane, titled '질의 결과' (Query Results), shows the execution results. It indicates that 1 row was returned in 0.027 seconds. The results are displayed in a table with the following columns and values:

	급여평균	급여합계	최대급여	최저급여
1	6461.831775700934579439252336448598130841	691416	24000	2100

**#Group By) 그룹으로 묶기★**

워크시트		질의 작성기																											
		<pre>SELECT job_id, AVG(salary) FROM employees GROUP BY job_id;</pre>																											
		<div> <div>계획 설명 x</div> <div>질의 결과 x</div> </div> <div>  SQL   인출된 모든 행: 19(0.006초)         </div> <table> <thead> <tr> <th></th><th>JOB_ID</th><th>AVG(SALARY)</th></tr> </thead> <tbody> <tr><td>1</td><td>IT_PROG</td><td>5760</td></tr> <tr><td>2</td><td>AC_MGR</td><td>12008</td></tr> <tr><td>3</td><td>AC_ACCOUNT</td><td>8300</td></tr> <tr><td>4</td><td>ST_MAN</td><td>7280</td></tr> <tr><td>5</td><td>PU_MAN</td><td>11000</td></tr> <tr><td>6</td><td>AD_ASST</td><td>4400</td></tr> <tr><td>7</td><td>AD_VP</td><td>17000</td></tr> <tr><td>8</td><td>SH_CLERK</td><td>3215</td></tr> </tbody> </table>		JOB_ID	AVG(SALARY)	1	IT_PROG	5760	2	AC_MGR	12008	3	AC_ACCOUNT	8300	4	ST_MAN	7280	5	PU_MAN	11000	6	AD_ASST	4400	7	AD_VP	17000	8	SH_CLERK	3215
	JOB_ID	AVG(SALARY)																											
1	IT_PROG	5760																											
2	AC_MGR	12008																											
3	AC_ACCOUNT	8300																											
4	ST_MAN	7280																											
5	PU_MAN	11000																											
6	AD_ASST	4400																											
7	AD_VP	17000																											
8	SH_CLERK	3215																											

//에러



3가지 기준 줘 보기



SQL 워크시트(W) | 내역

워크시트 | 질의 자동 추적... (F6)

```

SELECT job_id, AVG(salary), SUM(salary), COUNT(*)
FROM employees
GROUP BY job_id
ORDER BY AVG(salary) DESC;

```

계획 설명 x | 질의 결과 x

SQL | 인출된 모든 행: 19(0.007초)

	JOB_ID	AVG(SALARY)	SUM(SALARY)	COUNT(*)
1	AD_PRES	24000	24000	1
2	AD_VP	17000	34000	2
3	MK_MAN	13000	13000	1
4	SA_MAN	12200	61000	5
5	AC_MGR	12008	12008	1
6	FI_MGR	12008	12008	1
7	PU_MAN	11000	11000	1
8	HR_REP	10000	10000	1

♥응용♥

워크시트

질의 작성기

```
SELECT job_id, AVG(salary) 급여평균, SUM(salary) 급여합계, COUNT(*) 부서인원수
FROM employees
GROUP BY job_id
ORDER BY AVG(salary) DESC, COUNT(*) DESC;
```

계획 설명 x

질의 결과 x

SQL | 인출된 모든 행: 19(0.011초)

	JOB_ID	급여평균	급여합계	부서인원수
1	AD_PRES	24000	24000	1
2	AD_VP	17000	34000	2
3	MK_MAN	13000	13000	1
4	SA_MAN	12200	61000	5
5	AC_MGR	12008	12008	1
6	FI_MGR	12008	12008	1
7	PU_MAN	11000	11000	1
8	AD_PRES	10000	10000	1

-깔끔한 버전

워크시트		질의 작성기	
		<pre>SELECT job_id, AVG(salary) 급여평균, SUM(salary) 급여합계, COUNT(*) 부서인원수 FROM employees GROUP BY job_id ORDER BY 급여평균 DESC, 부서인원수 DESC;</pre>	

## #DML 데이터 조작 언어 Data Manipulation Language

SELECT 조회 read

INSERT 삽입 create

UPDATE 수정 update

DELETE 삭제 delete

\*CRUD = create, read, update, delete

## #DDL 데이터 정의 언어 Data Definition Language

DB 생성, Table 생성, 삭제 drop

## #DCL 데이터 제어 언어 Data Control Language

권한 관리

#새로운 테이블 만들기

CREATE TABLE 테이블이름

(열이름2 속성,

열이름2 속성

...

...);

CREATE TABLE board

(bno number,

btile varchar2(50),

bwriter varchar2(10),

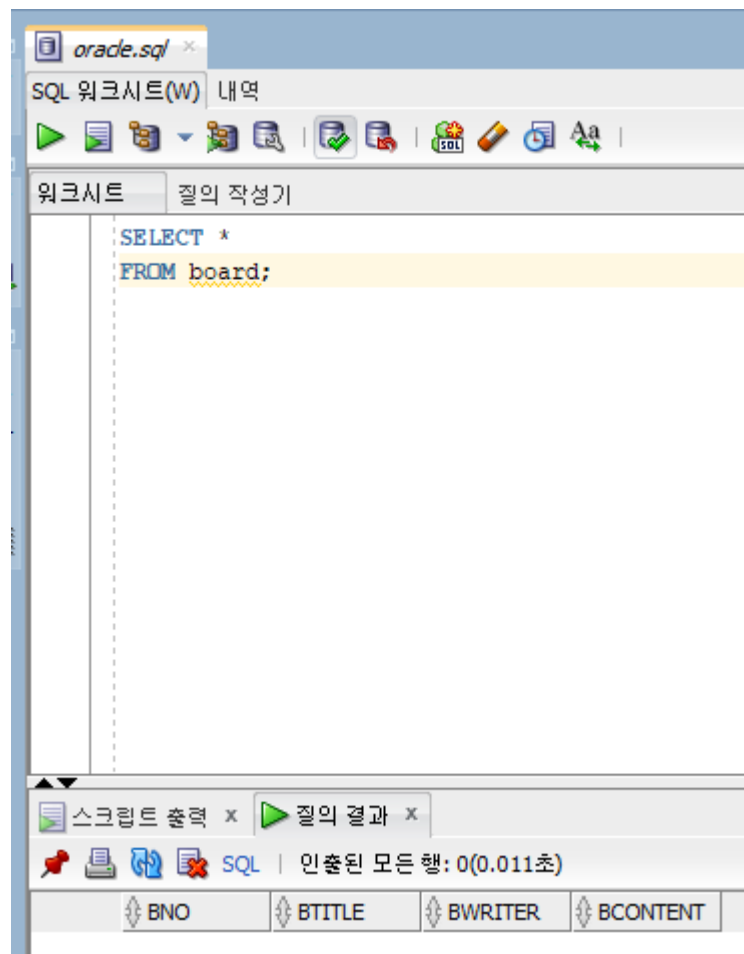
bcontent varchar2(500)

);

---

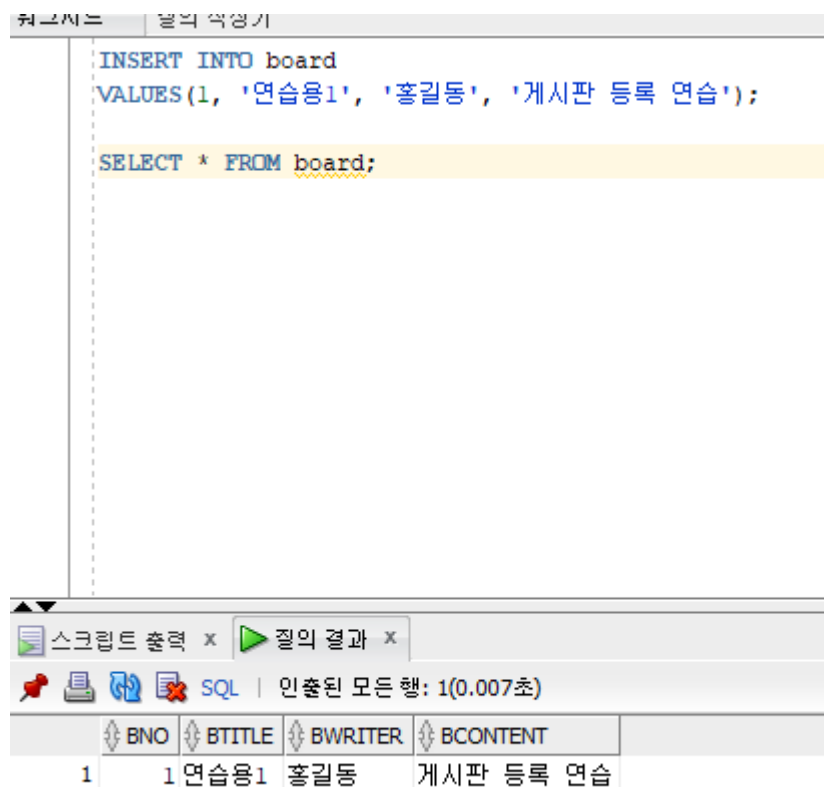
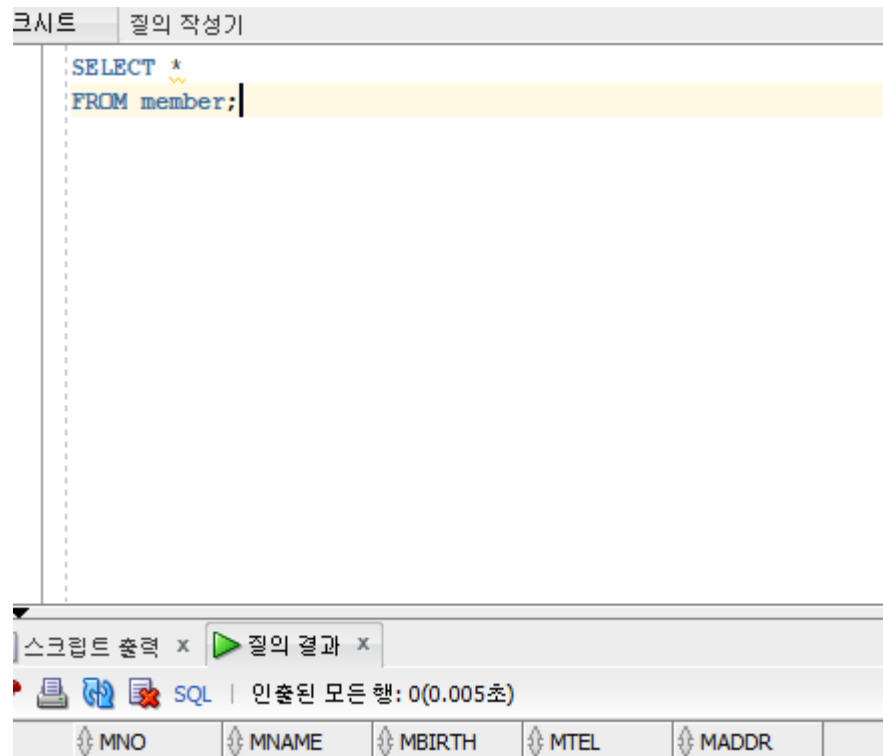
Table BOARD created.





```
CREATE TABLE member
(
  mno number,
  mname varchar2(50),
  mbirth varchar2(8),
  mtel varchar2(11),
  maddr varchar2(30)
);
```

Table MEMBER created.



## #이름 정의 방법

1. 동일한 이름의 테이블이 존재할 수 없다.
2. 예약어 즉 이미 사용 중인 명령어 등으로는 이름을 사용할 수 없다.
3. 반드시 문자로 시작해야 한다. 한글/특수문자도 쓸 수는 있지만 절대 사용하지 말자.
4. 가능하면 의미 있는 단어를 사용하자.

### #테이블 수정(항목 추가)

```
ALTER TABLE member  
ADD (mgender varchar2(10));
```

Table BOARD created.

Table MEMBER created.

1 row inserted.

Table MEMBER altered.

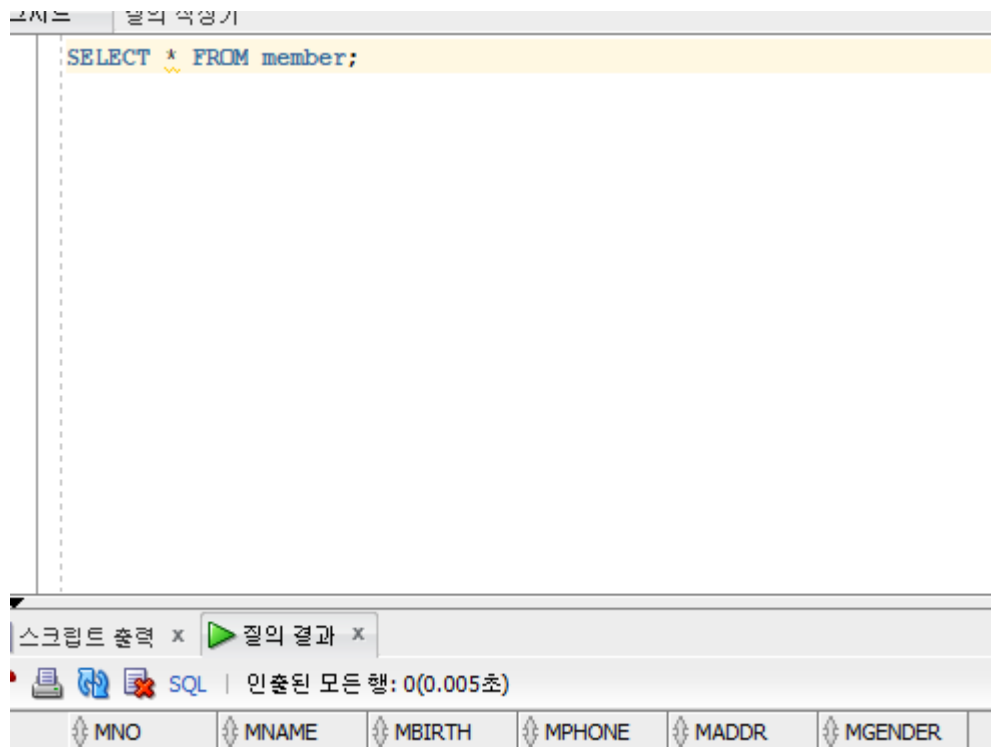
### #테이블 수정(항목 수)

```
ALTER TABLE member MODIFY (MTEL varchar2(20));
```

Table MEMBER altered.

```
ALTER TABLE member RENAME COLUMN MTEL to MPHONE;
```

Table MEMBER altered.



#테이블에서 전체 데이터 삭제(구조는 그대로)

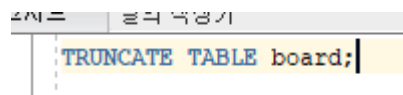
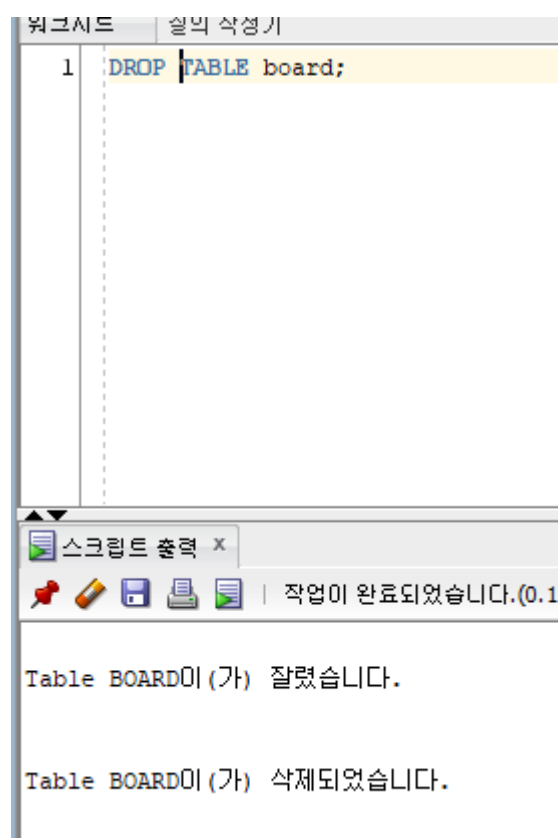
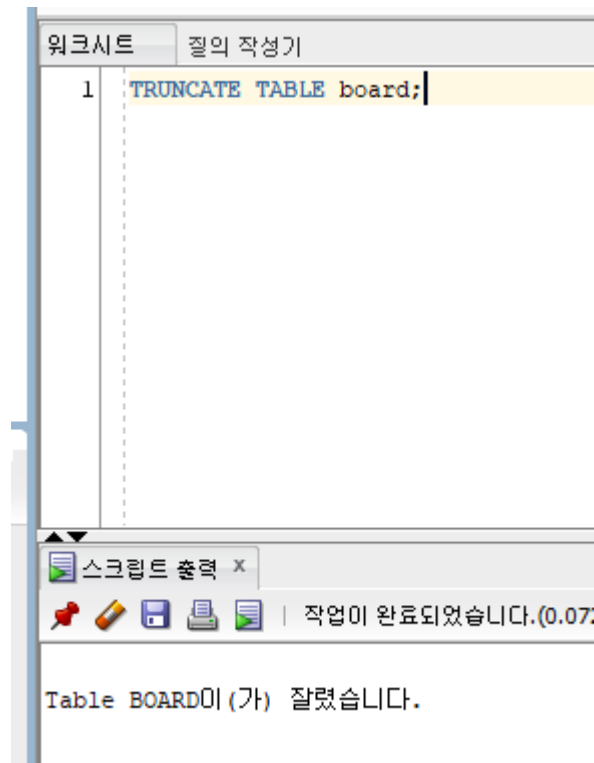
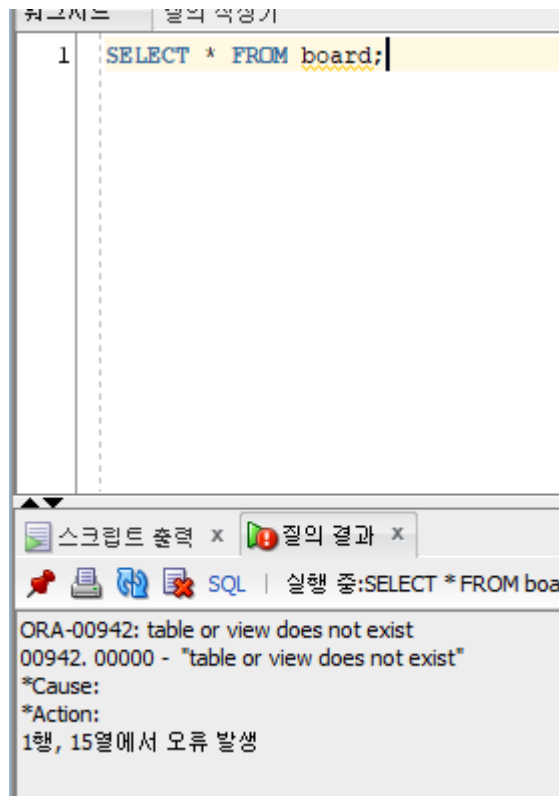


Table BOARD truncated.





### #삭제의 종류...

DML ) delete ) 데이터만 삭제

DDL ) truncate ) 구조를 남기고 데이터만 전체 삭제

DDL ) drop ) 구조 포함하여 데이터 전체 완전히 삭제

### #View ) 가상의 테이블

View는 데이터베이스에서 가상의 테이블이다. 실제로 테이블에 저장되어 있는 데이터를 그대로 사용하는 것이 아니라,

필요한 데이터만 추출하여 새로운 가상 테이블을 만들어서 사용한다. 뷰는 테이블과 동일하게 사용자에게 의해 생성되고, SQL문으로 조작이 가능하지만, 데이터는 뷰가 참조한 원본 테이블에 저장되어 있다.

뷰는 데이터를 중복해서 저장하지 않아도 되므로 데이터 정규화에 도움을 준다.

또한, 특정 사용자가 필요한 데이터만 조회하도록 제한하거나, 여러 개의 테이블에서 데이터를 조합하여 표시할 수 있다.



```
CREATE VIEW employee_view AS  
SELECT empoyee_id, first_name, last_name, salary, department_name  
FROM employees  
JOIN departments ON employees.department_id = departments.department_id;
```

### #뷰의 장점

뷰는 보안을 제공한다. 예를 들면, 보안 등급이 낮은 직원은 VIP 회원 테이블의 모든 정보를 다 볼 수 없도록 일부 항목만을 볼 수 있게 일부 항목들만 추출하여 뷰를 만들어 보안 등급이 낮은 직원이 해당 뷰만 볼 수 있도록 하면 보안상 이점을 확보할 수 있다.

-뷰는 데이터베이스에서 가상의 테이블이기 때문에, 보안상의 이점을 제공한다. 예를 들어, 보안 등급이 낮은 직원은 VIP 회원 테이블의 모든 정보를 볼 수 없도록

일부 항목들만 추출하여 새로운 뷰를 만들고 해당 뷰만 볼 수 있도록 하면, 보안성이 보장된다.

이를 통해, 뷰를 통해서만 필요한 정보에 접근하도록 설정하여 데이터베이스 전체의 보안성을 높일 수 있다.



worksheet    Query Builder

```
1 CREATE VIEW test_view AS
2 SELECT a.employee_id, a.hire_date, b.department_name, b.job_title
3 FROM employees A, emp_details_view B;
```

Query Result x    Script Output x

Task completed in 0.152 seconds

View TEST\_VIEW created.

Worksheet    Query Builder

```
1 SELECT *
2 FROM test_view;
```

Script Output x    Query Result x

SQL | Fetched 50 rows in 0.016 seconds

	EMPLOYEE_ID	HIRE_DATE	DEPARTMENT_NAME	JOB_TITLE
1	100	17-JUN-03	Marketing	Marketing Representative
2	100	17-JUN-03	Marketing	Marketing Manager
3	100	17-JUN-03	Public Relations	Public Relations Representative

