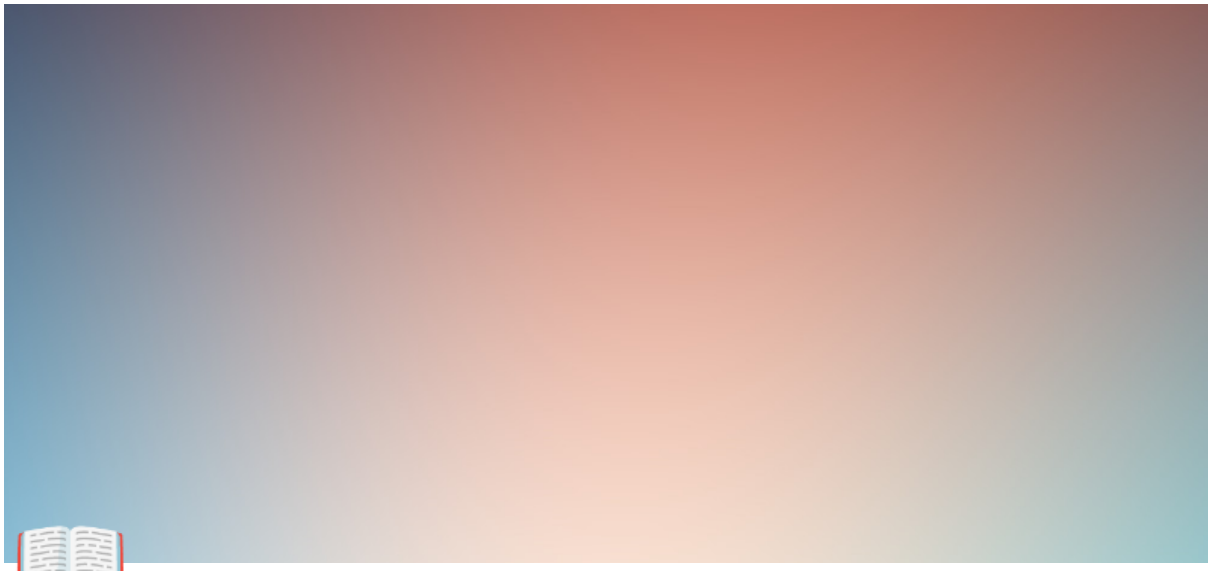
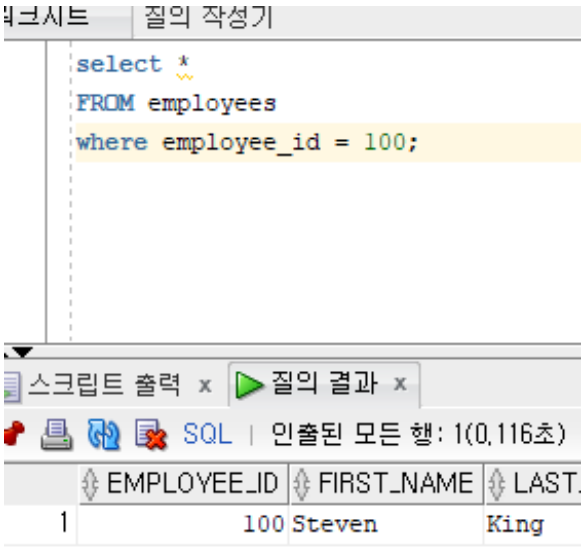


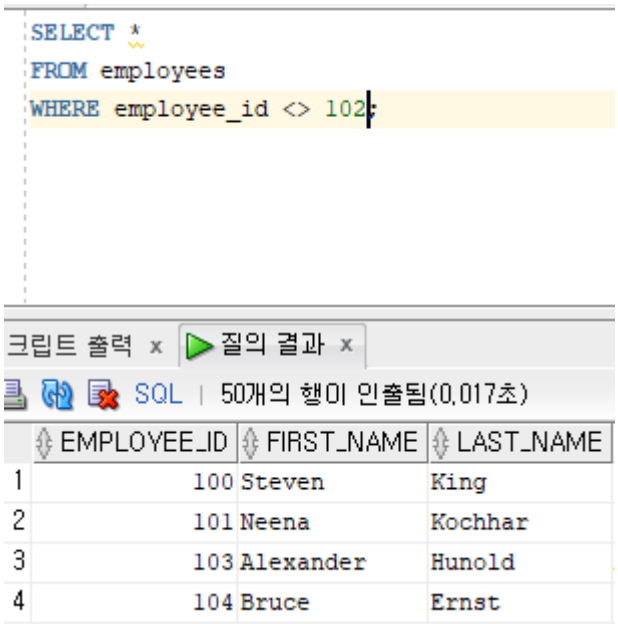
class_day_2

태그	
Dates	
Topic	
Type	
날짜	@2023년 5월 9일

- 비교 연산자
  - = 같다
  - != 다르다(<>)
  - > 크다
  - < 작다
  - <= 작거나 같다

  ▼ 실행 이미지





- 논리 연산자

  SELECT *
  FROM 테이블
  WHERE 조건1
  AND 조건2;

  - AND 모든 조건을 동시에 다 만족할 때만 true
  - OR 조건중 하나만 만족해도 true
  - NOT 조건의 반대 결과를 반환

▼ 실행 이미지

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

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 5(0.008초)

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

```sql
SELECT *
FROM employees
WHERE salary > 10000
OR job_id = 'IT_PROG';
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 20(0.021초)

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	03/06/17	AD_PRES	24000
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	05/09/21	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000
5	104	Bruce	Ernst	BERNST	590.423.4568	07/05/21	IT_PROG	6000
6	105	David	Austin	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800

```sql
SELECT *
FROM employees
WHERE salary > 10000
AND job_id = 'IT_PROG'
OR job_id = 'FI_ACCOUNT';
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 5(0.013초)

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
1	109	Daniel	Faviet	DFAVIET	515.124.4169	02/08/16	FI_ACCOUNT
2	110	John	Chen	JCHEN	515.124.4269	05/09/28	FI_ACCOUNT
3	111	Ismael	Sciarra	ISCIARRA	515.124.4369	05/09/30	FI_ACCOUNT
4	112	Jose Manuel	Urman	JMURMAN	515.124.4469	06/03/07	FI_ACCOUNT
5	113	Luis	Popp	LPOPP	515.124.4567	07/12/07	FI_ACCOUNT

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

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.003초)

	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
1	King	SKING	515.123.4567	03/06/17	AD_PRES	24000	(null)	(null)

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

크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.011초)

	ST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
1	ihar	NKOCHHAR	515.123.4568	05/09/21	AD_VP	17000	(null)	100
2	laan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000	(null)	100
3	ld	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	(null)	102

- 함수 사용

  ▼ 문자 관련 함수들

  - 단일 행 함수: 특정 행에 적용, 데이터 값을 하나씩 계산한다.

    1) 대문자/ 소문자/ 첫글자만 대문자 (BOY, boy, Boy)

▼ 실행 이미지

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

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.011초)

	LAST_NAME	LOWER(LAST_NAME)	UPPER(LAST_NAME)	EMAIL	INITCAP
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

INTCAP: 맨 앞자리만 대문자로 표기

2) 글자 자르기 (SUBSTR)

SUBSTR('원본 글자', 시작 위치, 자를 개수)

*SQL은 시작이 1부터 이다. (대부분 0부터 시작)

▼ 실행 이미지

```
SELECT job_id, SUBSTR(job_id, 1, 2)
FROM employees;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.016초)

	JOB_ID	SUBSTR(JOB_ID,1,2)
1	AC_ACCOUNT	AC
2	AC_MGR	AC
3	AD_ASST	AD
4	AD_PRES	AD
5	AD_VP	AD
6	AD_VP	AD
7	FI_ACCOUNT	FI
8	FI_ACCOUNT	FI

3) 글자 바꾸기 (replace) - 특정 문자를 찾아서 변경

REPLACE('문자열', '찾을문자', '바꿀문자')

▼ 실행 이미지

```
SELECT job_id, Replace(job_id,'ACCOUNT','ACC')
FROM employees;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.007초)

	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
7	FI_ACCOUNT	FI_ACC
8	FI_ACCOUNT	FI_ACC
9	FI_ACCOUNT	FI_ACC

*값이 바뀌진 않음

4) 특정 문자로 자리 채우기 (LPAD, RPAD)

왼쪽부터 채우기/ 오른쪽 채우기

LPAD('문자열', 만들어질 자리수, ''채울 문자'

▼ 실행 이미지

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

크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.009초)

	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
9	Sarah	*******Sarah
10	David	*******David

- 그룹 함수: 그룹 전체 적용, 여러개 값을 그룹으로 계산한다.

  1) 삭제하기 (LTRIM / RTRIM)

  LTRIM('문자열'/열이름, '삭제할 문자')

  ▼ 실행 이미지

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

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.007초)

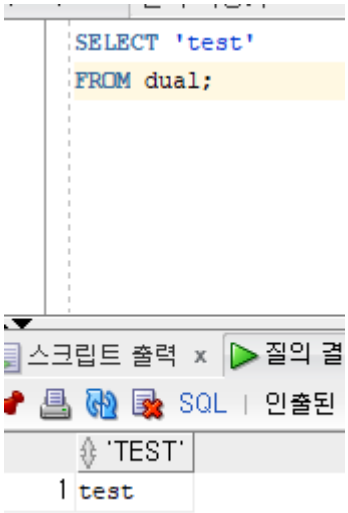
	JOB_ID	LTRIM(JOB_ID,'F')
7	FI_ACCOUNT	I_ACCOUNT
8	FI_ACCOUNT	I_ACCOUNT
9	FI_ACCOUNT	I_ACCOUNT
10	FI_ACCOUNT	I_ACCOUNT
11	FI_ACCOUNT	I_ACCOUNT
12	FI_MGR	I_MGR
13	HR_REP	HR_REP

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

스크립트 출력 x | 질의 결과 x

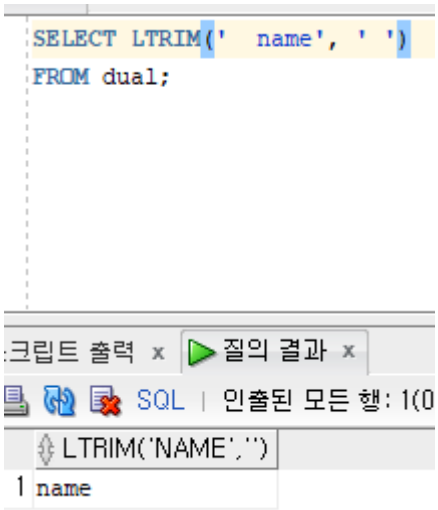
SQL | 50개의 행이 인출됨(0.008초)

	JOB_ID	LTRIM(JOB_ID,'F')	RTRIM(JOB_ID,'T')
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
8	FI_ACCOUNT	I_ACCOUNT	FI_ACCOUN
9	FI_ACCOUNT	I_ACCOUNT	FI_ACCOUN
10	FI_ACCOUNT	I_ACCOUNT	FI_ACCOUN
11	FI_ACCOUNT	I_ACCOUNT	FI_ACCOUN
12	FI_MGR	I_MGR	FI_MGR



dual 테이블은 dummy 테이블, 특정 테이블을 사용하지 않고 문법적으로 오류를 회피하고자 할 때 사용하는 일종이 가상의 테이블로 생각함.

*테이블이 없는 임의의 글자를 넣을떄 임시로 테이블에 넣을 수 있음(



▼ 숫자 관련 함수들

*소수점 아래로는 ~까지, 소수점 위로는 ~부터

1) 반올림(ROUND)

ROUND('열이름', 반올림할 위치)

▼ 실행 이미지

```
SELECT salary 월급, salary/30 일급, ROUND(salary/30, 0) 자리0, ROUND(salary/30, 1) "자리1 까지",
       ROUND(salary/30, -1) "자리1 부터"
FROM employees;
```

스크립트 출력 × 질의 결과 ×

SQL | 50개의 행이 인출됨(0.006초)

	월급	일급	자리0	자리1 까지	자리1 부터
1	24000	800	800	800	800
2	17000	566.6666666666666666666666666666666667	567	566.7	570
3	17000	566.6666666666666666666666666666666667	567	566.7	570
4	9000	300	300	300	300
5	6000	200	200	200	200
6	4800	160	160	160	160
7	4800	160	160	160	160
8	4200	140	140	140	140
9	12008	400.2666666666666666666666666666666667	400	400.3	400
10	9000	300	300	300	300
11	8200	273.3333333333333333333333333333333333	273	273.3	270
12	7700	256.6666666666666666666666666666666667	257	256.7	260
13	7800	260	260	260	260
14	6900	230	230	230	230
15	11000	366.6666666666666666666666666666666667	367	366.7	370
16	3100	103.3333333333333333333333333333333333	103	103.3	100

2)버림/절삭(TRUNC)

TRUNC(열이름, 자리값)

▼ 실행 이미지

```
SELECT salary 월급, salary/30 일급, TRUNC(salary/30, 0) 자리0, TRUNC(salary/30, 1) "자리1 까지",
       TRUNC(salary/30, -1) "자리1 부터"
FROM employees;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.006초)

	월급	일급	자리0	자리1 까지	자리1 부터
1	24000	800	800	800	800
2	17000	566.6666666666666666666666666666666667	566	566.6	560
3	17000	566.6666666666666666666666666666666667	566	566.6	560
4	9000	300	300	300	300
5	6000	200	200	200	200
6	4800	160	160	160	160
7	4800	160	160	160	160
8	4200	140	140	140	140
9	12008	400.2666666666666666666666666666666667	400	400.2	400
10	9000	300	300	300	300
11	8200	273.3333333333333333333333333333333333	273	273.3	270
12	7700	256.6666666666666666666666666666666667	256	256.6	250
13	7800	260	260	260	260

3) 데이터 형변환

- 자동 형변환(묵시적 형변환): 필요시 데이터 형을 자동으로 변환한다.

▼ 실행 이미지

```
SELECT 1 + '2'
FROM dual;
```

스크립트 출력 x | 질

SQL | 인

	1+'2'
1	3

숫자 2의 경우 작은 따옴표가 붙어있어 숫자가 아닌 문자다. 하지만 예외가 발생하지 않고 원하는 결과가 출력됐다. 이는 시스템이 자동으로 숫자로 변환해 계산한 것이다.

이렇듯 자동으로 형변환이 이루어 질 수 는 있지만 항상 사용자가 원하는 의도대로 변환이 자동으로 완벽하게 이루어지지 않는다. 따라서 자동 형변환이 잘 되더라도 수동으로 의도적인 명시를 해주는 것이 바람직 하다.

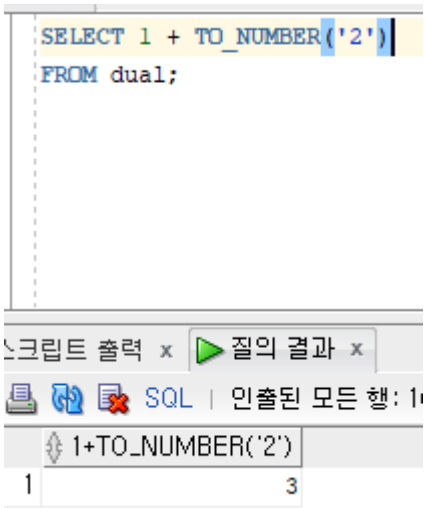
- 수동 형변환(명시적 형변환): 수동으로 데이터 형변환 한다.

  TO_CHAR 문자로 형변환

  TO_NUMBER 숫자로 형변환

  TO_DATE 날짜로 형변환

  ▼ 실행 이미지



VARCHAR2(varchar) —→NUMBER(integer)

NUMBER —→ VARCHAR2

▼ 기타 함수

1) NVL 함수

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

크립트 출력 x | 질의 결과 x

SQL | 100개의 행이 인출됨(0.007초)

	ST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
31	n	LDORAN	011.44.1345.629268	05/12/15	SA_REP	7500	0.3
32	r	JKING	011.44.1345.429268	04/01/30	SA_REP	10000	0.35
33	y	PSULLY	011.44.1345.929268	04/03/04	SA_REP	9500	0.35
34	en	AMCEWEN	011.44.1345.829268	04/08/01	SA_REP	9000	0.35
35	ell	JRUSSEL	011.44.1344.429268	04/10/01	SA_MAN	14000	0.4
36	r	SKING	515.123.4567	03/06/17	AD_PRES	24000	(null)
37	har	NKOCHHAR	515.123.4568	05/09/21	AD_VP	17000	(null)
38	laan	LDEHAAN	515.123.4569	01/01/13	AD_VP	17000	(null)
39	ld	AHUNOLD	590.423.4567	06/01/03	IT_PROG	9000	(null)
40	t	BERNST	590.423.4568	07/05/21	IT_PROG	6000	(null)
41	in	DAUSTIN	590.423.4569	05/06/25	IT_PROG	4800	(null)

salary와 commission_pct값을 계산하려고 하면 문제가 발생한다. 이유는 커미션을 지급받지 않는 직원 즉 null값이 있기 때문이다.

null로 계산하면 결과는 null이 된다.

null 값을 1로 정해서 계산하면 원래 값 그대로 나오기에 1로 치환해 계산하면 전체 계산에 문제를 발생시키지 않게된다.

NVL함수는 null값을 특정 값으로 치환하여 계산이 이루어지도록 처리한다.

▼ 실행 이미지

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

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 107(0.00

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

before

```
SELECT salary * NVL(commission_pct, 1)
FROM employees
ORDER BY commission_pct;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 107(0.006초)

	SALARY*NVL(COMMISSION_PCT,1)
29	3000
30	3300
31	2250
32	3500
33	3325
34	3150
35	5600
36	24000
37	17000
38	17000

after

2) 조건 처리하기(DECODE)

DECODE(열이름, 조건값, 치환값, 기본값)

- 치환값 - 조건을 만족할 경우
- 기본값 조건을 만족하지 않을 경우

▼ 실행 이미지

```
SELECT department_id,
       employee_id,
       first_name,
       salary "원래 급여",
       DECODE(department_id, 60, salary * 1.1, salary) "지급 급여",
       DECODE(department_id, 60, '급여 인상',' ') "인상 여부"
FROM employees;
```

크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.004초)

	DEPARTMENT_ID	EMPLOYEE_ID	FIRST_NAME	원래 급여	지급 급여	인상 여부
1	90	100	Steven	24000	24000	
2	90	101	Neena	17000	17000	
3	90	102	Lex	17000	17000	
4	60	103	Alexander	9000	9900	급여 인상
5	60	104	Bruce	6000	6600	급여 인상
6	60	105	David	4800	5280	급여 인상
7	60	106	Valli	4800	5280	급여 인상
8	60	107	Diana	4200	4620	급여 인상
9	100	108	Nancy	12008	12008	
0	100	109	Daniel	9000	9000	
1	100	110	John	8200	8200	
2	100	111	Ismael	7700	7700	

3) 복잡한 조건 처리(CASE, 경우의 수가 여러개일 경우)

▼ 실행 이미지

```sql
SELECT  job_id, employee_id, first_name, salary,
    CASE
        WHEN salary >= 9000 THEN '상급 개발자'
        WHEN salary >= 5000 THEN '중급 개발자'
        ELSE '하급 개발자'
    END AS 구분
FROM employees
WHERE job_id = 'IT_PROG';
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 5(0.002초)

	JOB_ID	EMPLOYEE_ID	FIRST_NAME	SALARY	구분
1	IT_PROG	103	Alexander	9000	상급 개발자
2	IT_PROG	104	Bruce	6000	중급 개발자
3	IT_PROG	105	David	4800	하급 개발자
4	IT_PROG	106	Valli	4800	하급 개발자
5	IT_PROG	107	Diana	4200	하급 개발자

- 순위 매기기 3가지 numbering
  - RANK 공통 순위 만큼 건너 뛰어 순위 매기기 1,2,2,4
  - DENSE_RANK 공통 순위를 건너 뛰지 않고 순위 1,2,2,3
  - ROW_NUMBER 공통순위 없이 출력 1,2,3,4

  ▼ 실행 이미지

```sql
SELECT employee_id, first_name, salary,
       RANK()OVER(ORDER BY salary DESC) "rank 순위",
       DENSE_RANK()OVER(ORDER BY salary DESC) "dense 순위",
       ROW_NUMBER()OVER(ORDER BY salary DESC) "row 순위"
FROM employees;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.005초)

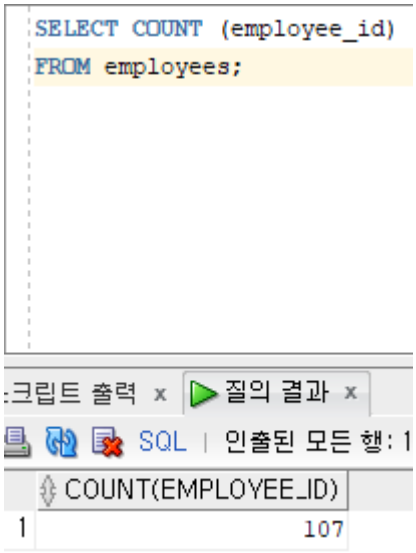
	EMPLOYEE_ID	FIRST_NAME	SALARY	rank 순위	dense 순위	row 순위
1	100	Steven	24000	1	1	1
2	101	Neena	17000	2	2	2
3	102	Lex	17000	2	2	3
4	145	John	14000	4	3	4
5	146	Karen	13500	5	4	5
6	201	Michael	13000	6	5	6
7	108	Nancy	12008	7	6	7
8	205	Shelley	12008	7	6	8
9	147	Alberto	12000	9	7	9
10	168	Lisa	11500	10	8	10

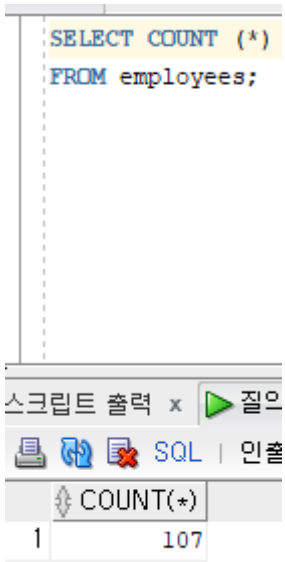
- 그룹 함수

  여러 행에 함수가 적용 되어 하나의 결과를 나타낸다.

  - count 갯수 (null값도 포함하여 샌다)

    ▼ 실행 이미지





count는 특성상 null도 계산하기 때문에 어떠한 열로도 동일한 결과 값이 나오기 때문에 *를 주로 사용한다.

- sum 합계 (null값 제외하고 계산)
- avg 평균 (null값 제외하고 계산)
- max 최댓값 (null값 제외하고 계산)
- min 최소값 (null값 제외하고 계산)

▼ 실행 이미지

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

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.002초)

	급여평균	급여합계	최대급여	최저급여
1	6461	691416	24000	2100

```
SELECT MAX(first_name) 최대값,
       MIN(first_name) 최저값
FROM employees;
```

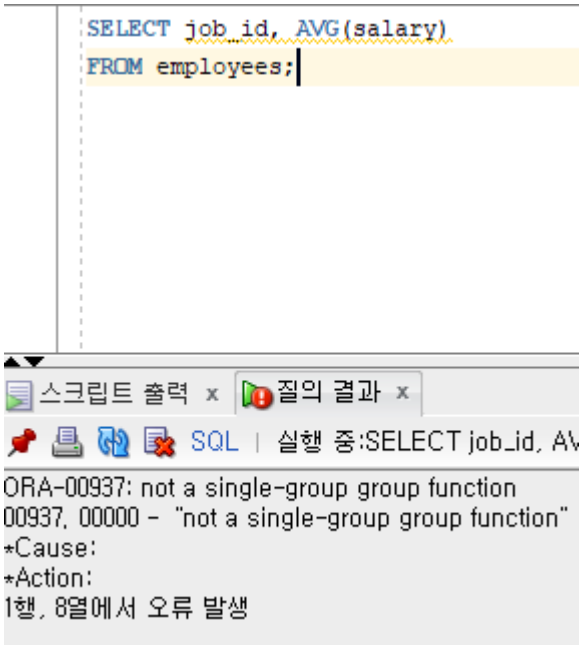
크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.0

	최대값	최저값
1	Winston	Adam

- 그룹으로 묶기(GROUP BY)

  ▼ 실행 이미지


SELECT job_id, AVG(salary)
FROM employees;
스크립트 출력
질의 결과
SQL
실행 중:SELECT job_id, AV
ORA-00937: not a single-group group function
00937, 00000 - "not a single-group group function"
*Cause:
*Action:
1행, 8열에서 오류 발생

before

```sql
SELECT job_id, AVG(salary)
FROM employees
GROUP BY job_id;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 19

	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
9	FI_ACCOUNT	7920
10	FI_MGR	12008
11	PU_CLERK	2780
12	SA_MAN	12200
13	MK_MAN	13000
14	PR_REP	10000
15	AD_PRES	24000
16	SA_REP	8350
17	MK_REP	6000
18	ST_CLERK	2785
19	HR_REP	6500

after

```sql
SELECT job_id, AVG(salary) "평균 월급", sum(salary) "월급 합", COUNT(*) "인원 수"
FROM employees
GROUP BY job_id
ORDER BY AVG(salary) DESC, COUNT(*) DESC;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 19(0.005초)

	JOB_ID	평균 월급	월급 합	인원 수
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	PR_REP	10000	10000	1
9	SA_REP	8350	250500	30
10	AC_ACCOUNT	8300	8300	1
11	FI_ACCOUNT	7920	39600	5
12	ST_MAN	7280	36400	5
13	HR_REP	6500	6500	1
14	MK_REP	6000	6000	1
15	IT_PROG	5760	28800	5
16	AD_ASST	4400	4400	1
17	SH_CLERK	3215	64300	20
18	ST_CLERK	2785	55700	20
19	PU_CLERK	2780	13900	5

▼ 데이터 명령어 종류

- DML: Data Manipulation Language = 데이터 조작어
  - SELECT 조회 read
  - INSERT 삽입 create
  - UPDATE 수정 update
  - DELETE 삭제 delete

CRUD로 정의

- DDL: Data Definition Language = 데이터 정의어

  DB 생성, Table 생성

  - CREATE
  - ALTER
  - DROP 삭제
  - RENAME
  - TRUNCATE
- DCL: Data Control Language = 데이터 제어어

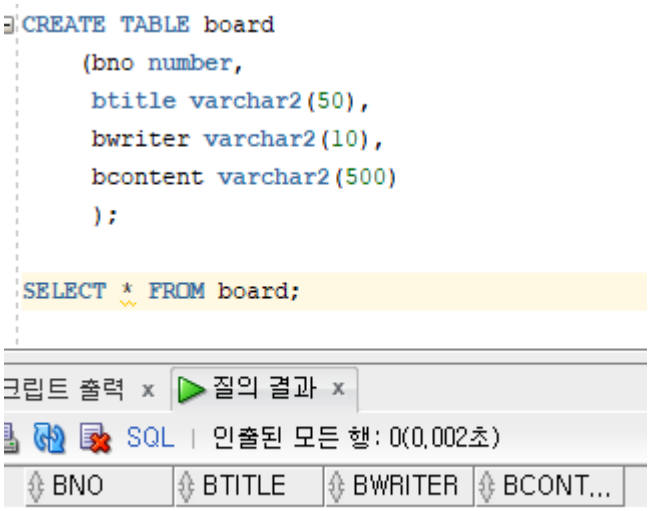
  권한 관리,

  - GRANT
  - REVOKE

- 새로운 테이블 만들기

  CREATE TABLE 테이블이름
  (열이름1 속성,
  열이름2 속성
  ...
  ...);

  ▼ 이미지 실행



```sql
CREATE TABLE member
    (mid number,
     mname varchar2(50),
     mold NUMBER,
     mbirth varchar2(8),
     mtel varchar2(11),
     maddr varchar2(300)
     );

select * from member;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 0(0.002초)

MID	MNAME	MOLD	MBIRTH	MTEL	MADDR

- 이름 정의 방법
  - 동일한 이름의 테이블 존재 불가
  - 예약어 즉 이미 사용 중인 명령어 등으로는 이름을 사용할 수 없다
  - 반드시 문자로 시작해야 한다. 한글 가능하나 사용하지 말 것
  - 가능하면 의미있는 단어 사용할 것

- 테이블 수정

  ▼ 항목 추가 (ADD)

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

select * from member;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.003초)

	MID	MNAME	MOLD	MBIRTH	MTEL	MADDR	MGENDER
1	1	길동 홍	10	200220	01000000000	우리집	(null)

▼ 항목 변경 (MODIFY, RENAME)

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

select * from member;
```

스크립트 출력 x | 질의 결과 x

작업이 완료되었습니

```
Table MEMBER이(가) 변경되었습니다.
```

MODIFY: 속성 변경

```
ALTER  TABLE member
RENAME COLUMN mtel TO mphone;

select * from member;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.004초)

	MID	MNAME	MOLD	MBIRTH	MPHONE	MADDR	MGENDER
1	1	길동 홍	10	200220	01000000000	우리집	(null)

RENAME COLUMN: 컬럼의 이름명 변경

▼ 열(컬럼) 삭제 (DROP)

테이블, 컬럼 등 삭제할때 쓰임

```
ALTER  TABLE member
DROP COLUMN maddr;

select * from member;
```

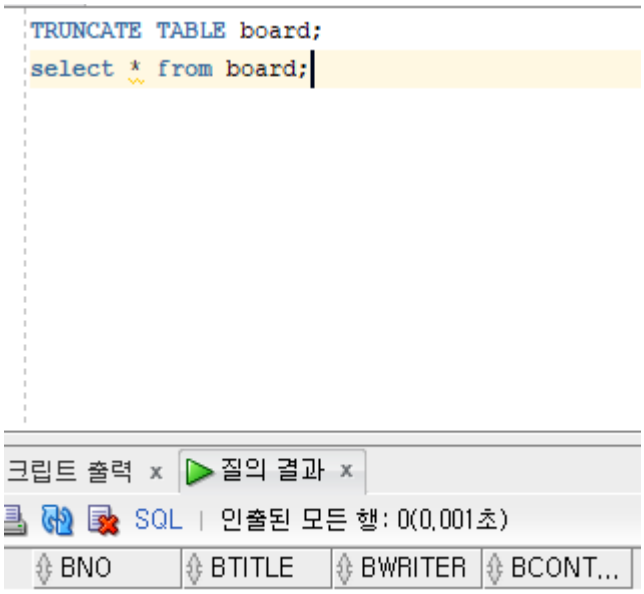
크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.002초)

	MID	MNAME	MOLD	MBIRTH	MPHONE	MGE...
1	1	길동 홍	10	200220	01000000000	(null)

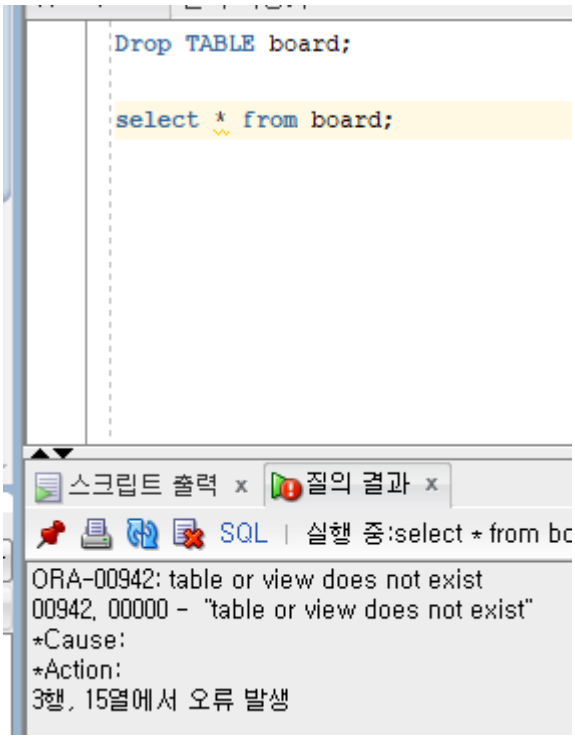
▼ 테이블에서 구조남기고 전체 데이터 삭제(TRUNCATE)

*구조는 그대로 남아있다.



내용만 지운 것, 인덱스까지 같이 삭제가 된다.

▼ 테이블 삭제 (DROP TABLE)



DML delete 데이터만 삭제

DDL truncate 구조를 남기고 데이터만 전체 삭제

DDL drop 구조 포함 전체 삭제

- 가상의 테이블(VIEW)
  - 실제로 테이블에 저장되어 있는 데이터를 그대로 사용하는 것이 아닌 필요 데이터만 추출해 새로운 가상 테이블을 만들어 사용한다. 뷰는 동일하게 사용자에 의해 생성되고 SQL문으로 조작 가능하지만, 데이터는 참조한 원본 테이블에 저장되어 있다. 뷰는 데이터 중복 저장이 불필요해 데이터 정규화에 도움을 준다. 또한 특정 사용자가 필요한 데이터만 조회하도록 제한하거나 여러 테이블의 데이터를 조합해 표시할 수 있다.

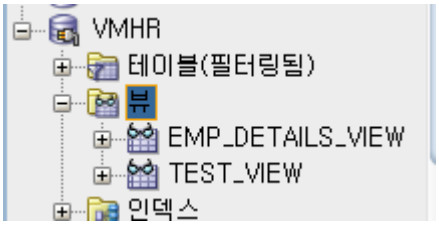
▼ 실행 이미지

```
SELECT A.employee_id, A.hire_date, B.department_name, b.job_title
FROM employees A, emp_details_view B;
```

스크립트 출력 x | 질의 결과 x

SQL | 50개의 행이 인출됨(0.016초)

	EMPLOYEE_ID	HIRE_DATE	DEPARTMENT_NAME	JOB_TITLE
1	100	03/06/17	Marketing	Marketing Representative
2	100	03/06/17	Marketing	Marketing Manager
3	100	03/06/17	Public Relations	Public Relations Representative
4	100	03/06/17	Sales	Sales Representative
5	100	03/06/17	Sales	Sales Representative
6	100	03/06/17	Sales	Sales Representative
7	100	03/06/17	Sales	Sales Representative
8	100	03/06/17	Sales	Sales Representative
9	100	03/06/17	Sales	Sales Representative
10	100	03/06/17	Sales	Sales Representative
11	100	03/06/17	Sales	Sales Representative
12	100	03/06/17	Sales	Sales Representative
13	100	03/06/17	Sales	Sales Representative
14	100	03/06/17	Sales	Sales Representative
15	100	03/06/17	Sales	Sales Representative
16	100	03/06/17	Sales	Sales Representative
17	100	03/06/17	Sales	Sales Representative
18	100	03/06/17	Sales	Sales Representative

  - 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 하여 보안상 이점을 확보할 수 있다.