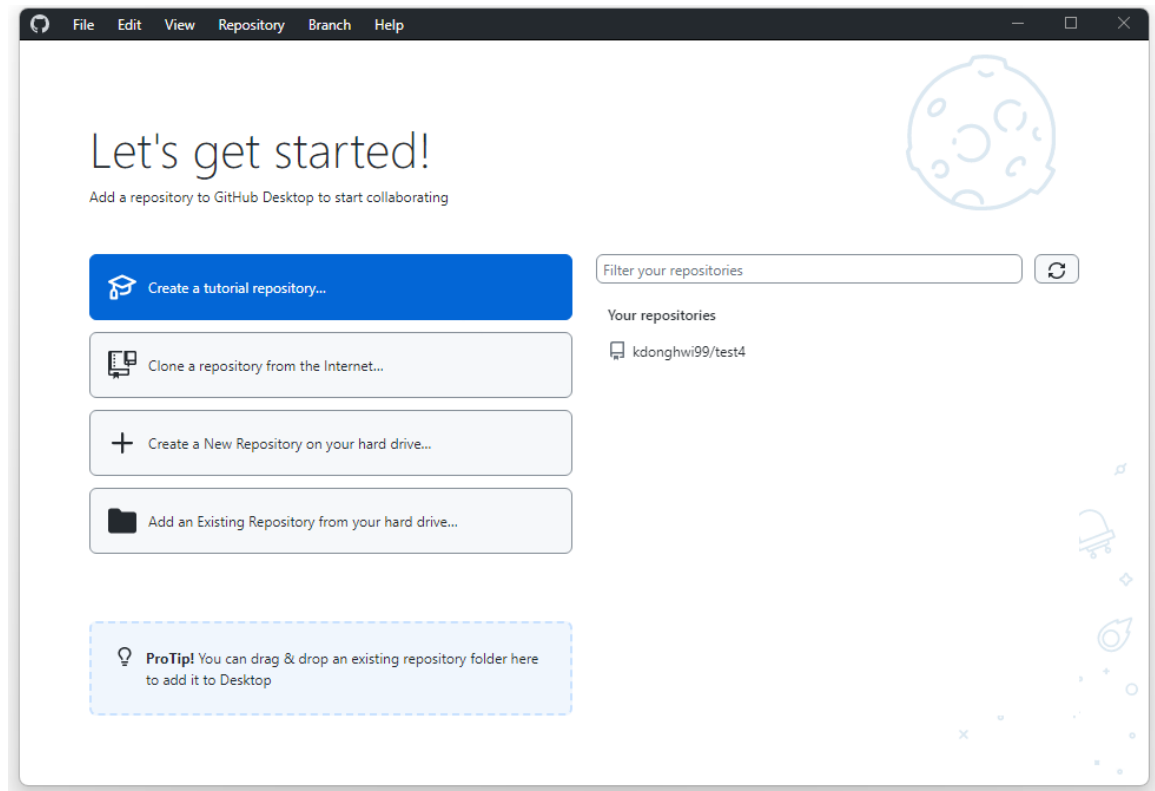
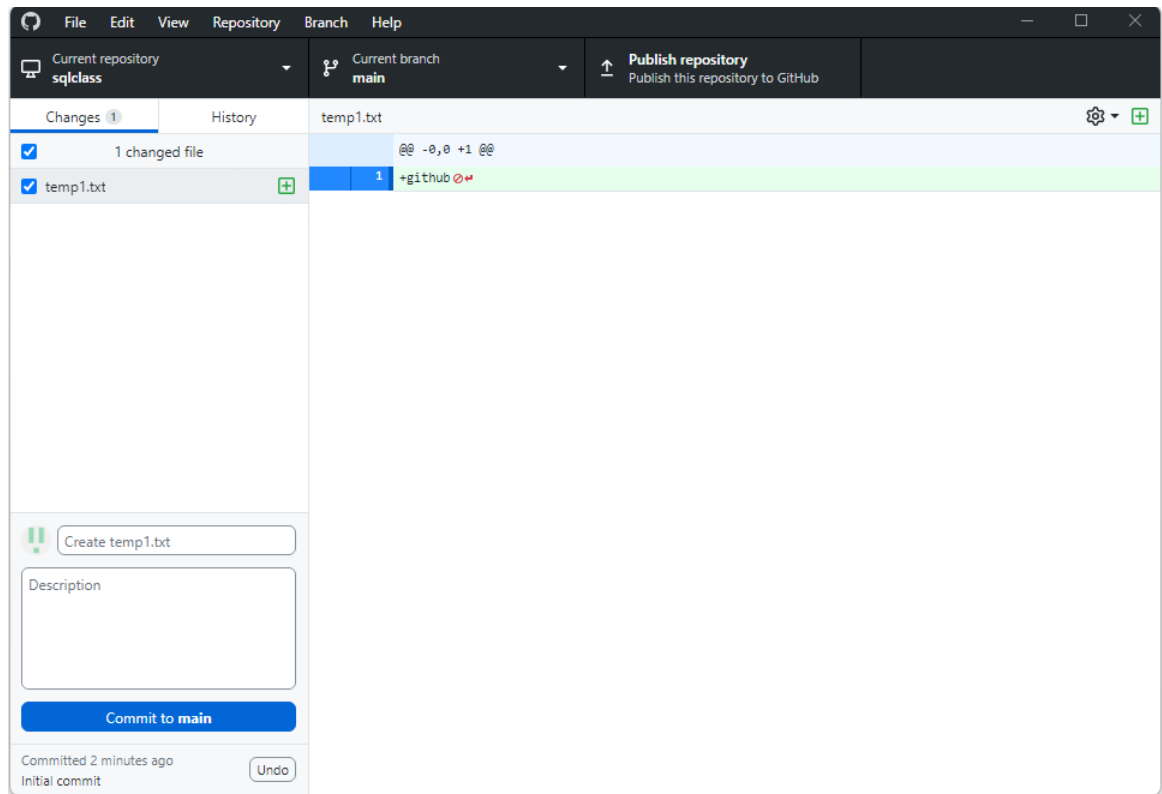


SQL CLASS 2

▼ 깃허브



Create a New Repository ⇒ Github에도 공간 생기고 컴퓨터에도 공간이 생겨 sink를 맞춤



실시간으로 생기는 파일을 감지하고 깃허브에 올릴 수 있음

▼ SQL

▼ 문자 관련 함수





- 비교 연산자(=, <>, !=, >, <, <=, >=)

Worksheet

Query Builder

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

Query Result x



SQL | All Rows Fetched: 1 in 1.338 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-03	AD PRES	24000

비교연산자 같다(=)를 이용하여 employee_id가 100인 행 조회

Worksheet

Query Builder

```
SELECT *  
FROM employees  
WHERE employee_id >= 200;
```

Query Result x

SQL | All Rows Fetched: 7 in 0.021 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-03	AD_ASST	4400
2	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000
3	202	Pat	Fay	PFAY	603.123.6666	17-AUG-05	MK_REP	6000
4	203	Susan	Mavris	SMAVRIS	515.123.7777	07-JUN-02	HR_REP	6500
5	204	Hermann	Baer	HBAER	515.123.8888	07-JUN-02	PR_REP	10000
6	205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-02	AC_MGR	12008
7	206	William	Gietz	WGIEZT	515.123.8181	07-JUN-02	AC_ACCOUNT	8300

비교연산자 크거나같다(>=)를 이용하여 employee_id가 200 이상인 행 조회

- 논리 연산자(AND, OR, NOT)

AND : 모든 조건을 동시에 만족할 경우 true

OR : 조건 중 하나만 만족할 경우 true

NOT : 조건의 반대 결과를 반환

Worksheet

Query Builder

```
SELECT *  
FROM employees  
WHERE salary > 4000  
AND job_id LIKE 'MK%';
```

Query Result x

SQL | All Rows Fetched: 2 in 0.024 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-04	MK_MAN	13000
2	202	Pat	Fay	PFAY	603.123.6666	17-AUG-05	MK_REP	6000


논리 연산자 AND 이용하여 salary가 4000 초과고 job_id가 MK로 시작하는 행 조회


```
SELECT *
FROM employees
WHERE salary > 4000
AND job_id LIKE 'MK%';
```

Worksheet

Query Builder

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

 Query Result x

 All Rows Fetched: 10 in 0.023 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000
2	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	6000
3	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	4800
4	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	4800
5	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4200
6	109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-02	FI_ACCOUNT	9000
7	110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	8200
8	111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-05	FI_ACCOUNT	7700

논리 연산자 AND와 OR 동시 사용

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

Worksheet

Query Builder

```
SELECT *  
FROM employees  
WHERE employee_id <> 105;
```

Query Result x

SQL | Fetched 50 rows in 0.032 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
1	100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG
6	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG
7	107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG

논리 연산자 NOT을 이용하여 employee_id가 105가 아닌 행 조회





```
SELECT *
FROM employees
WHERE employee_id <> 105;
```

Worksheet

Query Builder

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

Query Result x

    SQL | All Rows Fetched: 1 in 0.01 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	24000

manager_id에 NULL값이 있는 행 조회

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

Worksheet

Query Builder

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

Query Result x

SQL

| Fetched 50 rows in 0.024 seconds

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
1	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP
2	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	AD_VP
3	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG

논리 연산자 NOT을 이용하여 manager_id에 NULL값이 없는 행들을 조회(employee_id 100 제외)

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

- 함수 사용

1) 단일 행 함수 : 특정 행에 적용, 데이터 값을 하나씩 계산

1. 대문자/소문자/첫글자만 대문자(오라클에만 존재, MySQL에는 존재x)

Worksheet

Query Builder

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

Query Result x

SQL | Fetched 50 rows in 0.017 seconds



	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

대문자로 변환(UPPER), 소문자로 변환(LOWER), 첫글자만 대문자로 변환(INITCAP)

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

2. 글자 자르기(substr)

문법 - **SUBSTR('원본글자', 시작위치, 자를 개수)** ※ **SQL은 1부터 시작** ※

Worksheet		Query Builder																								
		<pre>SELECT job_id, SUBSTR(job_id,1, 4) FROM employees;</pre>																								
		<div>  Query Result x </div> <div>  SQL Fetched 50 rows in 0.0 </div> <table> <thead> <tr> <th></th><th>JOB_ID</th><th>SUBSTR(JOB_ID,1,4)</th></tr> </thead> <tbody> <tr><td>1</td><td>AC_ACCOUNT</td><td>AC_A</td></tr> <tr><td>2</td><td>AC_MGR</td><td>AC_M</td></tr> <tr><td>3</td><td>AD_ASST</td><td>AD_A</td></tr> <tr><td>4</td><td>AD_PRES</td><td>AD_P</td></tr> <tr><td>5</td><td>AD_VP</td><td>AD_V</td></tr> <tr><td>6</td><td>AD_VP</td><td>AD_V</td></tr> <tr><td>7</td><td>FI_ACCOUNT</td><td>FI_A</td></tr> </tbody> </table>		JOB_ID	SUBSTR(JOB_ID,1,4)	1	AC_ACCOUNT	AC_A	2	AC_MGR	AC_M	3	AD_ASST	AD_A	4	AD_PRES	AD_P	5	AD_VP	AD_V	6	AD_VP	AD_V	7	FI_ACCOUNT	FI_A
	JOB_ID	SUBSTR(JOB_ID,1,4)																								
1	AC_ACCOUNT	AC_A																								
2	AC_MGR	AC_M																								
3	AD_ASST	AD_A																								
4	AD_PRES	AD_P																								
5	AD_VP	AD_V																								
6	AD_VP	AD_V																								
7	FI_ACCOUNT	FI_A																								

1번째 글자부터 4번째 글자까지 자르기

```
SELECT job_id,
SUBSTR(job_id,1, 4)
FROM employees;
```

3. 글자 바꾸기 - 특정 문자를 찾아서 변경

문법 - **REPLACE('문자열', '찾을 문자', '바꿀 문자')** ※ 임시로 변경해서 보여줌 ※

Worksheet		Query Builder
		<pre>SELECT job_id, REPLACE(job_id, 'ACCOUNT', 'AC') FROM employees;</pre>
		Query Result x
		SQL Fetched 50 rows in 0.032 seconds
	JOB_ID	REPLACE(JOB_ID,'ACCOUNT','AC')
1	AC_ACCOUNT	AC_AC
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_AC
8	FI_ACCOUNT	FI_AC

REPLACE 이용하여 ACCOUNT를 AC로 줄임

```
SELECT job_id,
REPLACE(job_id, 'ACCOUNT', 'AC')
FROM employees;
```

4. 특정 문자로 자리 채우기(LPAD, RPAD)

ex - 개인정보 보호시 주민번호에 *로 자리 채움

문법 - **LPAD('문자열', 만들 자리수, '채울 문자')**

Worksheet		Query Builder
		<pre>SELECT first_name, LPAD(first_name, 12, '*') FROM employees;</pre>
		Query Result x
		SQL Fetched 50 rows in 0.019 sec
	FIRST_NAME	LPAD(FIRST_NAME,12,'*')
1	Ellen	*****Ellen
2	Sundar	*****Sundar
3	Mozhe	*****Mozhe
4	David	*****David
5	Hermann	*****Hermann
6	Shelli	*****Shelli

LPAD 이용하여 왼쪽부터 *로 채움


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

문법 - **RPAD('문자열', 만들 자리수, '채울 문자')**

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

RPAD 이용하여 오른쪽부터 *로 채움

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

2) 그룹 함수 : 그룹 전체에 적용, 여러개 값을 그룹으로 계산

1. 삭제 (STRIM, RTRIM)

문법 - **LTRIM('문자열'/열이름, '삭제할 문자')**

문법 - **RTRIM('문자열'/열이름, '삭제할 문자')**

Worksheet

Query Builder

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

2. 임의의 테이블(dual)

dual 테이블이란 dummy 테이블, 특정 테이블을 사용하지 않고 문법적으로 오류를 회피하고자 할 때 사용하는 일종의 가상 테이블

Worksheet		Query Builder	
		<pre>SELECT LTRIM(' name', ' ') FROM dual;</pre>	
Query Result ×			
		SQL All Rows Fetched: 1 in 0.0	
		LTRIM('NAME',")	
1	name		

```
SELECT LTRIM(' name', ' ')
FROM dual;
```

▼ 숫자 관련 함수

1. 반올림(round)

문법 - **ROUND(열이름, 자리값)**

Worksheet

Query Builder

ROUND 이용하여 여러 자리에서 반올림 실행

```
SELECT salary AS 월급,  
salary/30 AS 일급,  
ROUND(salary/30, 2) AS 소수점둘째,  
ROUND(salary/30, 0) AS 소수점x,  
ROUND(salary/30, -1) AS 일의자리올림,  
ROUND(salary/30, -2) AS 십의자리올림  
FROM employees;
```

2. 버림(trunc)

문법 - **TRUNC(열이름, 자리값)**

Worksheet

Query Builder

SELECT salary AS 월급,
salary/30 AS 일급,
TRUNC(salary/30, 2) AS 소수점두번째버림,
TRUNC(salary/30, 0) AS 자연수,
TRUNC(salary/30, -1) AS 일의자리버림
FROM employees;

Query Result x

SQL | Fetched 50 rows in 0.034 seconds

	월급	일급	소수점두번째버림	자연수	일의자리버림
1	24000	800	800	800	800
2	17000	566.6666...	566.66	566	560
3	17000	566.6666...	566.66	566	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	400.2666...	400.26	400	400

TRUNC 이용하여 여러 자리에서 버림 실행

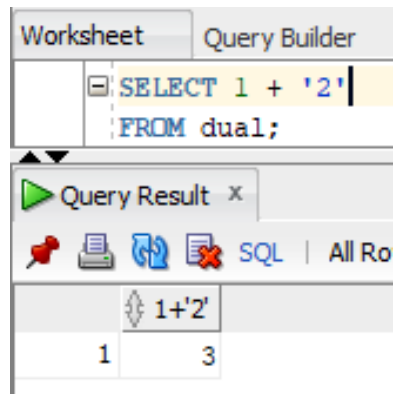
```
SELECT salary AS 월급,
salary/30 AS 일급,
TRUNC(salary/30, 2) AS 소수점두번째버림,
TRUNC(salary/30, 0) AS 자연수,
TRUNC(salary/30, -1) AS 일의자리버림
FROM employees;
```

3. 데이터 형변환(casting)

자동 형변환(묵시적 형변환) - 필요시 데이터 형을 자동으로 변환

VARCHAR2(문자) ↔ NUMBER(숫자) (Oracle)

VARCHAR(문자) ↔ INTEGER(숫자) (My SQL)



'2' 자동 형변환(문자 → 숫자)

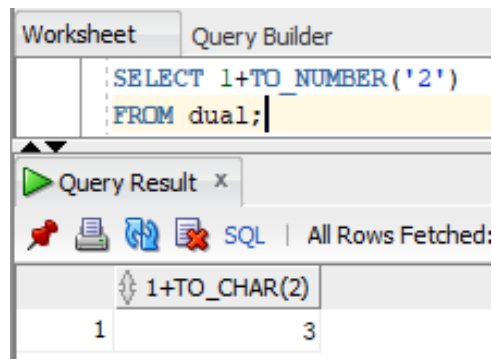
```
SELECT 1+'2'
FROM dual;
```

수동 형변환(명시적 형변환) - 수동으로 데이터 형변환

TO_CHAR : 문자로 형변환

TO_NUMBER : 숫자로 형변환

TO_DATE : 날짜로 형변환



TO_NUMBER 수동 형변환(문자 → 숫자)

```
SELECT 1+TO_NUMBER('2')
FROM dual;
```

▼ 기타 함수

1) 단일 행 함수





1. null값 처리(NVL)

Worksheet

Query Builder

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

Query Result x

    SQL | All Rows Fetched: 107 in 0.022 seconds

	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID
30	GCAMBRAU	011.44.1344.619268	15-OCT-07	SA_MAN	11000	0.3	100
31	LDORAN	011.44.1345.629268	15-DEC-05	SA_REP	7500	0.3	146
32	JKING	011.44.1345.429268	30-JAN-04	SA_REP	10000	0.35	146
33	PSULLY	011.44.1345.929268	04-MAR-04	SA_REP	9500	0.35	146
34	AMCEWEN	011.44.1345.829268	01-AUG-04	SA_REP	9000	0.35	146
35	JRUSSEL	011.44.1344.429268	01-OCT-04	SA_MAN	14000	0.4	100
36	SKING	515.123.4567	17-JUN-03	AD_PRES	24000	(null)	(null)
37	NKOCHHAR	515.123.4568	21-SEP-05	AD_VP	17000	(null)	100
38	LDEHAAN	515.123.4569	13-JAN-01	AD_VP	17000	(null)	100
39	AHUNOLD	590.423.4567	03-JAN-06	IT_PROG	9000	(null)	102

commission_pct에 있는 null값 확인





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

Worksheet

Query Builder

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

Query Result x

    SQL | Fetched 50 rows in 0.009 seconds

	SALARY	COMMISSION_PCT	SALARY*COMMISSION_PCT
30	11000	0.3	3300
31	7500	0.3	2250
32	10000	0.35	3500
33	9500	0.35	3325
34	9000	0.35	3150
35	14000	0.4	5600
36	24000	(null)	(null)
37	17000	(null)	(null)
38	17000	(null)	(null)
39	9000	(null)	(null)

salary와 commission_pct를 계산할 경우 커미션을 지급받지 않은 직원(null)이 존재하기 때문에 문제 발생 ⇒ null 계산시 결과도 null이 됨

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

문법 - NVL(열이름, 치환할 값)

Worksheet

Query Builder

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

Query Result x

SQL | Fetched 50 rows in 0.017 seconds

	SALARY	NVL(COMMISSION_PCT,1)	SALARY*NVL(COMMISSION_PCT,1)
30	11000	0.3	3300
31	7500	0.3	2250
32	10000	0.35	3500
33	9500	0.35	3325
34	9000	0.35	3150
35	14000	0.4	5600
36	24000	1	24000
37	17000	1	17000
38	17000	1	17000
39	9000	1	9000

null값을 1로 치환하여 계산하면 전체 계산에 문제를 발생시키지 않음

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

2. 조건 처리(DECODE) ⇒ JAVA의 IF문과 사

문법 - DECODE(열이름, 조건값, 치환값, 기본값)






치환값 → 조건을 만족할 경우 / 기본값 → 조건을 만족하지 않을 경우

Worksheet

Query Builder

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

Query Result x

     SQL | Fetched 50 rows in 0.007 seconds

	DEPARTMENT_ID	EMPLOYEE_ID	FIRST_NAME	원래급여	DECODE...
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
8	60	107	Diana	4200	4620
9	100	108	Nancy	12008	12008
10	100	109	Daniel	9000	9000
11	100	110	John	8200	8200

department_id가 60이면 급여인상(salary*1.1), 그렇지 않으면 그대로 지급

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





3. 복잡한 조건 처리(CASE) ⇒ 조건이 여러개

Worksheet

Query Builder

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

Query Result x

    SQL | All Rows Fetched: 5 in 0.009 seconds

	JOB_ID	EMPLOYEE_ID	FIRST_NAME	SALARY	등급
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

salary가 9000 이상일 경우 상급개발자, salary가 5000 이상일 경우 중급개발자, 그렇지 않으면 하급개발자 출력

```

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

```

4. 순위 매기기(RANK, DENSE_RANK, ROW_NUMBER)

RANK : 공통 순위만큼 건너 뛰어 순위 매기기 ex - 1 → 2 → 2 → 4 → 5 → ...

DENSE_RANK : 공통 순위를 건너 뛰지 않고 순위 매기기 ex -
1 → 2 → 2 → 3 → 4 → ...

ROW_NUMBER : 공통 순위 없이 순위 매기기 ex - 1 → 2 → 3 → 4 → 5 → ...

Worksheet		Query Builder																																																					
		<pre>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;</pre>																																																					
		<div>Query Result x</div> <div>SQL Fetched 50 rows in 0.014 seconds</div> <table> <tr> <th></th><th>EMPLOYEE_ID</th><th>FIRST_NAME</th><th>SALARY</th><th>RANK순위</th><th>DENSE순위</th><th>ROW순위</th></tr> <tr><td>1</td><td>100</td><td>Steven</td><td>24000</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>2</td><td>101</td><td>Neena</td><td>17000</td><td>2</td><td>2</td><td>2</td></tr> <tr><td>3</td><td>102</td><td>Lex</td><td>17000</td><td>2</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>145</td><td>John</td><td>14000</td><td>4</td><td>3</td><td>4</td></tr> <tr><td>5</td><td>146</td><td>Karen</td><td>13500</td><td>5</td><td>4</td><td>5</td></tr> <tr><td>6</td><td>201</td><td>Michael</td><td>13000</td><td>6</td><td>5</td><td>6</td></tr> </table>						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
	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																																																	

순위 매기기 방법 3가지 비교

```
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;
```

2) 그룹함수

1. count, sum, avg, max, min

COUNT 개수 / SUM 합계 / AVG 평균 / MAX 최댓값 / MIN 최솟값





※ COUNT는 null 값을 포함하여 계산하지만 나머지는 null 값을 제외하고 계산 ※

Worksheet

Query Builder

```
SELECT COUNT(*)  
FROM employees;
```

▶ Query Result x



SQL | All Rows Fetched

	COUNT(*)
1	107

count 이용하여 행의 개수 세기

count는 특성상 null값을 포함하여 계산하기 때문에 어떠한 열로도 동일한 결과가 나옴

⇒ count(*)을 주로 사용

```
SELECT COUNT(*)  
FROM employees;
```

The screenshot shows the SQL Developer interface. The 'Query Builder' tab is active, displaying a query: `SELECT ROUND(AVG(salary), 1) 급여평균, SUM(salary) 급여합계, MAX(salary) 최대급여, MIN(salary) 최저급여 FROM employees;`. Below the query, the 'Query Result' window shows the results of the query. The results are as follows:

	급여평균	급여합계	최대급여	최저급여
1	6461.8	691416	24000	2100

sum, avg, max, min 사용

```
SELECT ROUND(AVG(salary), 1) 급여평균,  
SUM(salary) 급여합계,  
MAX(salary) 최대급여,  
MIN(salary) 최저급여  
FROM employees;
```

The screenshot shows the SQL Developer interface. The 'Query Builder' tab is active, displaying a query: `SELECT MAX(first_name) 최대값, MIN(first_name) 최소값 FROM employees;`. Below the query, the 'Query Result' window shows the results of the query. The results are as follows:

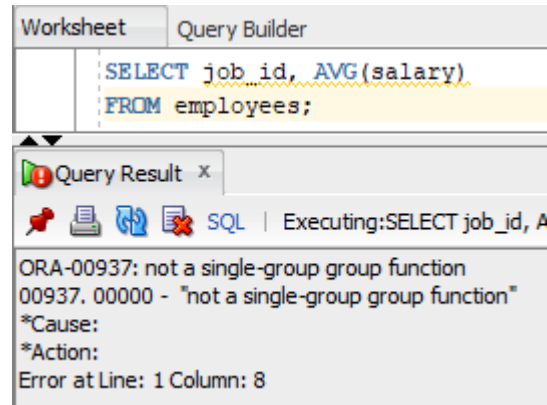
	최대값	최소값
1	Winston	Adam

문자열의 최대, 최소

문자열을 ASCII 코드로 바꾸어서 최대, 최소값을 구할 수 있음

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

2. 그룹 묶기(GROUP BY)



group으로 묶지 않은 경우 오류 발생

Worksheet		Query Builder																																																									
		<pre>SELECT job_id, AVG(salary) 급여평균, SUM(salary) 급여합계, COUNT(*) 부서인원수 FROM employees GROUP BY job_id ORDER BY 급여평균 DESC;</pre>																																																									
		<p>Query Result x</p> <p>SQL All Rows Fetched: 19 in 0.009 seconds</p> <table> <thead> <tr> <th></th><th>JOB_ID</th><th>급여평균</th><th>급여합계</th><th>부서인원수</th></tr> </thead> <tbody> <tr><td>1</td><td>AD_PRES</td><td>24000</td><td>24000</td><td>1</td></tr> <tr><td>2</td><td>AD_VP</td><td>17000</td><td>34000</td><td>2</td></tr> <tr><td>3</td><td>MK_MAN</td><td>13000</td><td>13000</td><td>1</td></tr> <tr><td>4</td><td>SA_MAN</td><td>12200</td><td>61000</td><td>5</td></tr> <tr><td>5</td><td>AC_MGR</td><td>12008</td><td>12008</td><td>1</td></tr> <tr><td>6</td><td>FI_MGR</td><td>12008</td><td>12008</td><td>1</td></tr> <tr><td>7</td><td>PU_MAN</td><td>11000</td><td>11000</td><td>1</td></tr> <tr><td>8</td><td>PR_REP</td><td>10000</td><td>10000</td><td>1</td></tr> <tr><td>9</td><td>SA_REP</td><td>8350</td><td>250500</td><td>30</td></tr> <tr><td>10</td><td>AC_ACCOUNT</td><td>8300</td><td>8300</td><td>1</td></tr> </tbody> </table>				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	PR_REP	10000	10000	1	9	SA_REP	8350	250500	30	10	AC_ACCOUNT	8300	8300	1
	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	PR_REP	10000	10000	1																																																							
9	SA_REP	8350	250500	30																																																							
10	AC_ACCOUNT	8300	8300	1																																																							

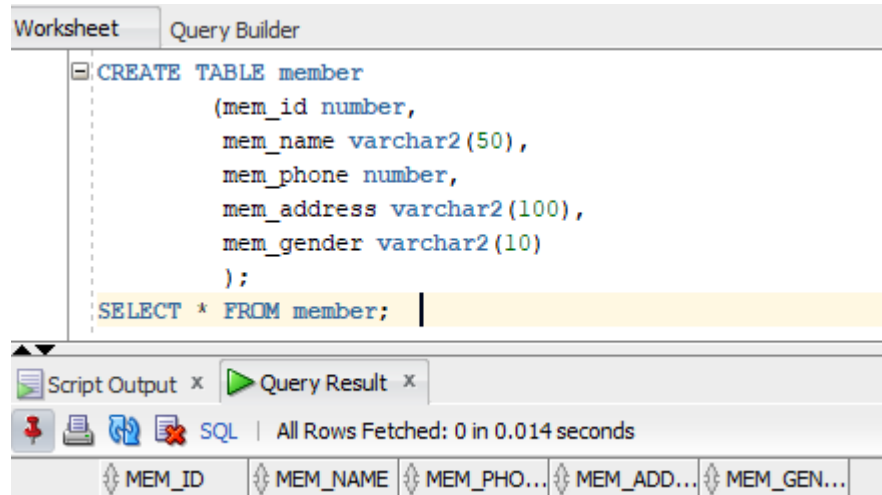
group으로 묶어서 job_id별 salary의 평균, 합계, 개수 조회

```
SELECT job_id,
AVG(salary) 급여평균, SUM(salary) 급여합계, COUNT(*) 부서인원수
FROM employees
```

```
GROUP BY job_id  
ORDER BY 급여평균 DESC;
```

▼ 테이블 직접 만져보기★

1. 테이블 생성(create)



create 사용하여 table 직접 만들기

```
CREATE TABLE member  
(  
    mem_id      number,  
    mem_name    varchar2(50),  
    mem_phone   number,  
    mem_address varchar2(100),  
    mem_gender  varchar2(10)  
);  
SELECT * FROM member;
```

2. 테이블에 값 대입

Worksheet Query Builder

```

INSERT INTO member VALUES (1, '홍길동', 99998888, '마포구', 'M');
INSERT INTO member VALUES (2, '이영희', 12345678, '강남구', 'F');
INSERT INTO member VALUES (3, '김유나', 22233883, '구로구', 'F');
INSERT INTO member VALUES (4, '박다영', 38571938, '강서구', 'F');
INSERT INTO member VALUES (5, '이철수', 98473724, '구로구', 'M');

SELECT * FROM member;

```

Script Output x Query Result x Query Result 1 x

SQL | All Rows Fetched: 5 in 0.011 seconds

	MEM_ID	MEM_NAME	MEM_PHONE	MEM_ADDRESS	MEM_GENDER
1	1	홍길동	99998888	마포구	M
2	2	이영희	12345678	강남구	F
3	3	김유나	22233883	구로구	F
4	4	박다영	38571938	강서구	F
5	5	이철수	98473724	구로구	M

새로 만든 member 테이블에 값 넣기

```

INSERT INTO member VALUES (1, '홍길동', 99998888, '마포구', 'M');
INSERT INTO member VALUES (2, '이영희', 12345678, '강남구', 'F');
INSERT INTO member VALUES (3, '김유나', 22233883, '구로구', 'F');
INSERT INTO member VALUES (4, '박다영', 38571938, '강서구', 'F');
INSERT INTO member VALUES (5, '이철수', 98473724, '구로구', 'M');

```

3. 테이블 내용 변경

```

ALTER TABLE member MODIFY (mem_name varchar2(10));
ALTER TABLE member RENAME COLUMN mem_gender to mem_gen;
ALTER TABLE member DROP COLUMN mem_phone;

```

Script Output x Query Result x Query Result 1 x

SQL | All Rows Fetched: 5 in 0.022 seconds

	MEM_ID	MEM_NAME	MEM_ADDRESS	MEM_GEN
1	1	홍길동	마포구	M
2	2	이영희	강남구	F
3	3	김유나	구로구	F
4	4	박다영	강서구	F
5	5	이철수	구로구	M

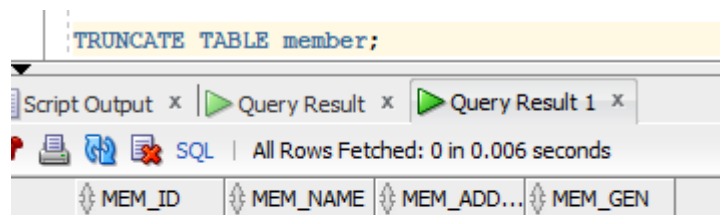
mem_name의 속성 변경, mem_gender의 이름 변경, mem_phone의 삭제

```
ALTER TABLE member MODIFY (mem_name varchar2(10));
ALTER TABLE member RENAME COLUMN mem_gender to mem_gen;
ALTER TABLE member DROP COLUMN mem_phone;
```

4. 데이터, 테이블 삭제

※ 데이터 삭제 → DELETE(DML)

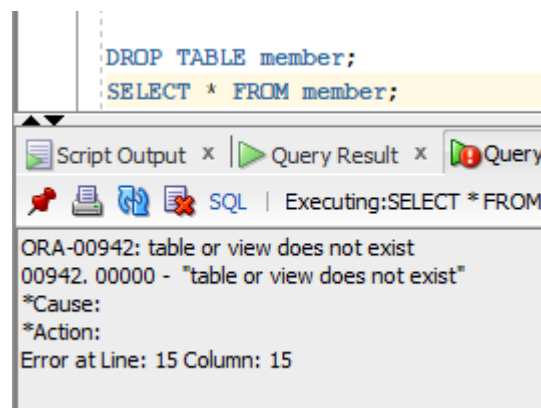
※ 전체 데이터 삭제(구조 유지) → **TRUNCATE**(DDL)



truncate 사용하여 전체 데이터 삭제

```
TRUNCATE TABLE member;
```

※ 전체 데이터 삭제(구조 포함) → **DROP**(DDL)



drop을 사용하여 테이블 삭제
⇒ select 했을 경우 오류 발생

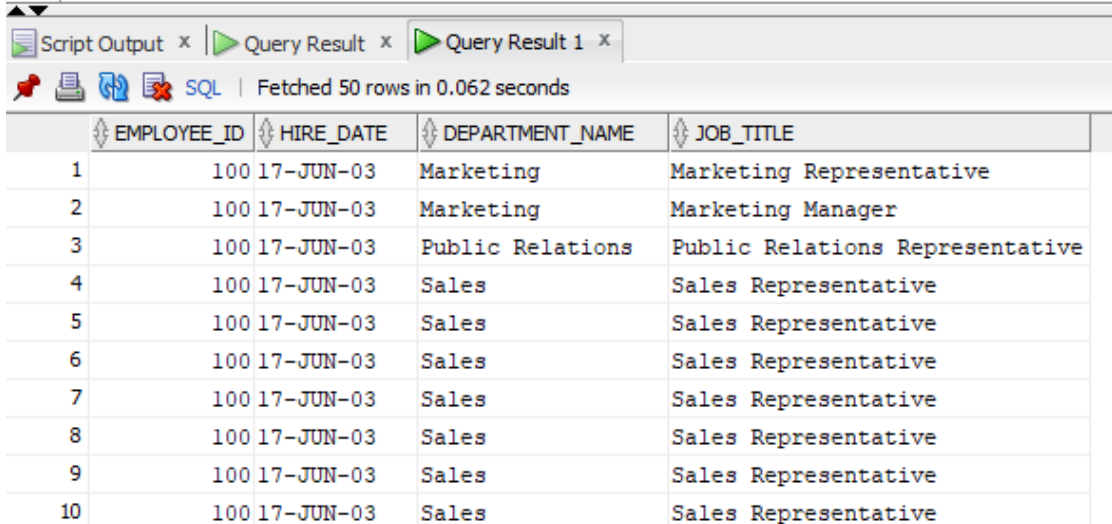
```
DROP TABLE member;
SELECT * FROM member;
```

5. 가상의 테이블(VIEW)

실제로 테이블에 저장되어있는 데이터를 그대로 사용하는 것이 아닌, 필요한 데이터만 추출하여 새로운 가상 테이블을 만들어서 조작함

(장) 보안 제공, ex - 보안 등급이 낮은 직원은 VIP 회원 테이블의 모든 정보를 다 볼 수 없고 일부 항목만을 볼 수 있게 추출하여 뷰를 생성 ⇒ 보안 등급이 낮은 직원은 해당 뷰만 볼 수 있도록 하여 보안성 높임

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



	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
4	100	17-JUN-03	Sales	Sales Representative
5	100	17-JUN-03	Sales	Sales Representative
6	100	17-JUN-03	Sales	Sales Representative
7	100	17-JUN-03	Sales	Sales Representative
8	100	17-JUN-03	Sales	Sales Representative
9	100	17-JUN-03	Sales	Sales Representative
10	100	17-JUN-03	Sales	Sales Representative

employees의 별명을 A, emp_details_view의 별명을 B으로 설정하고
A에서 가져온 employee_id, hire_date, B에서 가져온 department_name, job_title 속성 조회

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

※ 이름 정의 방법

- 동일한 이름의 테이블은 존재할 수 없음
- 예약어, 즉 이미 사용 중인 명령어 등의 이름을 사용할 수 없음
- 반드시 문자로 시작. 한글/특수문자도 가능하지만 가능한 사용x
- 가능하면 의미있는 단어를 사용

▼ 데이터 언어

- 데이터 언어 종류

DML(Data Manipulation Language) : 데이터 조작 언어

INSERT 삽입(Create) / **SELECT** 조회(Read) / **UPDATE** 수정(Update) /
DELETE 삭제>Delete)

⇒ CRUD

DDL(Data Definition Language) : 데이터 정의 언어

DB 생성, TABLE 생성, 삭제(drop)

DCL(Data Control Language) : 데이터 제어 언어

권한 부여 grant, 권한 취소revoke