

SELECT문장 ROWNUM, ROWID 데이터 제한 데이터 정렬



Selection : 질의에 대해 리턴하고자 하는 테이블의 행을 선택하기 위해 SQL의 selection 기능을 사용할 수 있습니다. 보고자 하는 행을 선택적으로 제한하기 위해 다양한 방법을 사용할 수 있습니다.

Join : 공유 테이블 양쪽의 열에 대해 링크를 생성하여 다른 테이블에 저장되어 있는 데이터를 함께 가져오기 위해 SQL의 join 기능을 사용할 수 있습니다.

1. Selection

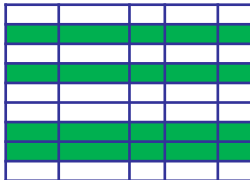
A 10x5 grid representing a table. The 2nd, 3rd, 4th, and 7th rows are highlighted in green, indicating they are the selected data.

Table 1

2. Join

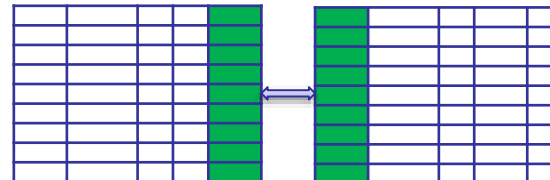
Two 10x5 grids representing tables. The 5th column of Table 1 and the 1st column of Table 2 are highlighted in green. A double-headed arrow connects these two columns, indicating a join operation.

Table 1

Table 2



```
SELECT [DISTINCT] { * | column [[AS] alias], ... }  
FROM table;
```

구문형식에서...

- SELECT : 하나 이상의 열을 나열합니다.
- DISTINCT : 중복을 제거합니다.
- * : 모든 열을 선택합니다.
- column : 명명된 열을 선택합니다.
- AS : 열 별칭(alias)을 지정합니다.
- alias : 선택된 열을 다른 이름으로 변경합니다.
- FROM table : 열을 포함하는 테이블을 명시합니다.

SQL 문장은 대/소문자를 구별하지 않습니다.

SQL 문장은 한 줄 이상일 수 있습니다.

키워드는 단축하거나 줄을 나누어 쓸 수 없습니다.

절은 대개 줄을 나누어서 씁니다.

탭과 들여쓰기(indent)는 읽기 쉽게 하기 위해 사용됩니다.

```
SQL> SELECT first_name, last_name, salary  
2 FROM employees;
```

	FIRST_NAME	LAST_NAME	SALARY
1	Steven	King	24000
2	Neena	Kochhar	17000
3	Lex	De Haan	17000
4	Alexander	Hunold	9000
5	Bruce	Ernst	6000
6	David	Austin	4800
7	Valli	Pataballa	4800
8	Diana	Lorentz	4200

```
SQL> SELECT *
      2 FROM departments;
```

질의 결과 x

SQL | 인출된 모든 행: 27(0.003초)

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	30	Purchasing	114	1700
4	40	Human Resources	203	2400
5	50	Shipping	121	1500
6	60	IT	103	1400
7	70	Public Relations	204	2700
8	80	Sales	145	2500
생략				
21	210	IT Support	(null)	1700
22	220	NOC	(null)	1700
23	230	IT Helpdesk	(null)	1700
24	240	Government Sales	(null)	1700
25	250	Retail Sales	(null)	1700
26	260	Recruiting	(null)	1700
27	270	Payroll	(null)	1700

질의 결과 x

SQL | 인출된 모든 행: 27(0.002초)

DEPARTMENT_NAME	LOCATION_ID
1 Administration	1700
2 Marketing	1800
3 Purchasing	1700
4 Human Resources	2400
5 Shipping	1500
6 IT	1400
7 Public Relations	2700
8 Sales	2500
9 Executive	1700
10 Finance	1700
11 Accounting	1700
12 Treasury	1700
13 Corporate Tax	1700
14 Control And Credit	1700
15 Shareholder Services	1700
16 Benefits	1700
17 Manufacturing	1700
18 Construction	1700
19 Contracting	1700
20 Operations	1700
21 IT Support	1700
22 NOC	1700
23 IT Helpdesk	1700
24 Government Sales	1700
25 Retail Sales	1700
26 Recruiting	1700
27 Payroll	1700

```
SQL> SELECT department_name, location_id
2 FROM departments;
```

```
SQL> SELECT location_id, department_name
2 FROM departments;
```

질의 결과 x

SQL | 인출된 모든 행: 27(0.003초)

LOCATION_ID	DEPARTMENT_NAME
1	1700 Administration
2	1800 Marketing
3	1700 Purchasing
4	2400 Human Resources
5	1500 Shipping
6	1400 IT
7	2700 Public Relations
8	2500 Sales
9	1700 Executive
10	1700 Finance
11	1700 Accounting
12	1700 Treasury
13	1700 Corporate Tax
14	1700 Control And Credit
15	1700 Shareholder Services
16	1700 Benefits
17	1700 Manufacturing
18	1700 Construction
19	1700 Contracting
20	1700 Operations
21	1700 IT Support
22	1700 NOC
23	1700 IT Helpdesk
24	1700 Government Sales
25	1700 Retail Sales
26	1700 Recruiting
27	1700 Payroll

디폴트 데이터 자리맞춤을 지정합니다.
 날짜와 문자 데이터는 왼쪽 정렬됩니다.
 숫자 데이터는 오른쪽 정렬됩니다.
 디폴트 열은 대문자로 출력됩니다.

```
SQL> SELECT first_name, hire_date, salary
2 FROM employees;
```

질의 결과 x

SQL | 인출된 모든 행: 107(0,011초)

	FIRST_NAME	HIRE_DATE	SALARY
1	Steven	03/06/17	24000
2	Neena	05/09/21	17000
3	Lex	01/01/13	17000
4	Alexander	06/01/03	9000
5	Bruce	07/05/21	6000
6	David	05/06/25	4800
7	Valli	06/02/05	4800
8	Diana	07/02/07	4200
9	Nancy	02/08/17	12008
10	Daniel	02/08/16	9000
11	John	05/09/28	8200
12	Ismael	05/09/30	7700
13	Jose Manuel	06/03/07	7800
14	Luis	07/12/07	6900
15	Den	02/12/07	11000
16	Alexander	03/05/18	3100
17	Shelli	05/12/24	2900

...

곱하기와 나누기는 더하기와 빼기보다 우선순위가 높습니다.

같은 우선순위의 연산자는 좌측에서 우측으로 계산됩니다.

괄호는 강제로 계산의 우선순위를 바꾸거나 문장을 명료하게 하기 위해 사용됩니다

$\ast / + -$

```
SQL> SELECT first_name, last_name, salary, salary+salary*0.1
2 FROM employees;
```

질의 결과 x

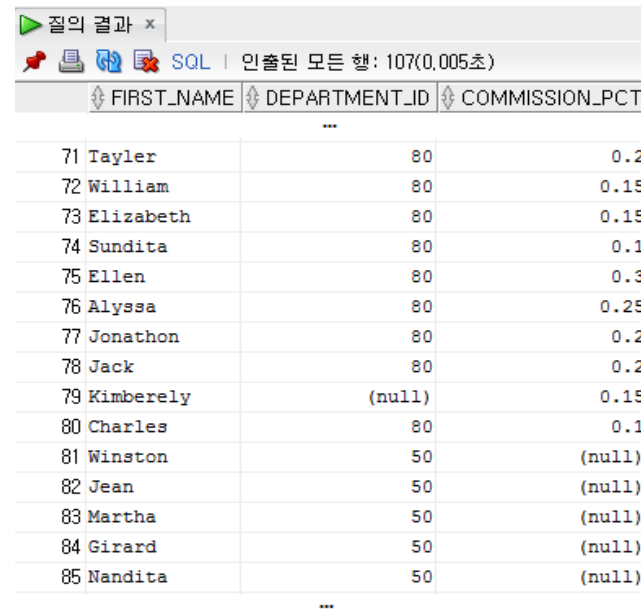
SQL | 인출된 모든 행: 107(0,008초)

	FIRST_NAME	LAST_NAME	SALARY	SALARY+SALARY*0.1
1	Steven	King	24000	26400
2	Neena	Kochhar	17000	18700
3	Lex	De Haan	17000	18700
4	Alexander	Hunold	9000	9900
5	Bruce	Ernst	6000	6600
6	David	Austin	4800	5280
7	Valli	Pataballa	4800	5280
8	Diana	Lorentz	4200	4620
9	Nancy	Greenberg	12008	13208.8
10	Daniel	Faviet	9000	9900
		...		

Null은 이용할 수 없거나, 지정되지 않았거나, 알 수 없거나 또는 적용할 수 없는 값입니다.

Null은 숫자 0이나 공백과는 다릅니다.

```
SQL> SELECT first_name, department_id, commission_pct  
2 FROM employees;
```



	FIRST_NAME	DEPARTMENT_ID	COMMISSION_PCT
71	Taylor	80	0.2
72	William	80	0.15
73	Elizabeth	80	0.15
74	Sundita	80	0.1
75	Ellen	80	0.3
76	Alyssa	80	0.25
77	Jonathon	80	0.2
78	Jack	80	0.2
79	Kimberely	(null)	0.15
80	Charles	80	0.1
81	Winston	50	(null)
82	Jean	50	(null)
83	Martha	50	(null)
84	Girard	50	(null)
85	Nandita	50	(null)

열 헤딩 이름을 변경합니다.

계산할 때에 유용합니다.

열 이름 바로 뒤에 둡니다. 열 이름과 별칭 사이에 키워드 'AS'를 넣기도 합니다.

공백이나 특수문자 또는 대/소문자가 있으며 이중 인용부호(" ")가 필요합니다.

```
SQL> SELECT first_name AS 이름, salary 급여
2 FROM employees;
```

	이름	급여
1	Steven	24000
2	Neena	17000

```
SQL> SELECT first_name "Employee Name",
2 salary*12 "Annual Salary"
3 FROM employees;
```

	Employee Name	Annual Salary
1	Steven	288000
2	Neena	204000

SELECT 절에 포함된 리터럴은 문자 표현식 또는 숫자입니다.

날짜와 문자 리터럴 값은 단일 인용부호(' ')안에 있어야 합니다.

숫자 리터럴은 단일 인용부호(' ')를 사용하지 않습니다.

각각의 문자스트링은 리턴된 각 행에 대한 결과입니다.

||를 이용하면 값을 연결해 줍니다.

```
SQL> SELECT first_name || ' ' || last_name || ''s salary is $' || salary  
2      AS "Employee Details"  
3 FROM      employees;
```

	Employee Details
1	Steven King's salary is \$24000
2	Neena Kochhar's salary is \$17000
3	Lex De Haan's salary is \$17000
4	Alexander Hunold's salary is \$9000
5	Bruce Ernst's salary is \$6000
6	David Austin's salary is \$4800
7	Valli Pataballa's salary is \$4800
8	Diana Lorentz's salary is \$4200
9	Nancy Greenberg's salary is \$12008
10	Daniel Faviyet's salary is \$9000
11	John Chen's salary is \$8200
12	Ismael Sciarra's salary is \$7700

의의 디폴트 출력은 중복되는 행을 포함하는 모든 행입니다.

SELECT 절에서 **DISTINCT** 키워드를 사용하여 **중복되는 행을 제거**합니다.

```
SQL> SELECT department_id
       2 FROM employees;
```

	DEPARTMENT_ID
1	90
2	90
3	90
4	60
5	60
6	60
7	60
8	60
9	100
10	100
11	100
12	100
13	100
14	100
15	30
16	30
17	30

...

```
SQL> SELECT DISTINCT department_id
       2 FROM employees;
```

	DEPARTMENT_ID
1	100
2	30
3	(null)
4	90
5	20
6	70
7	110
8	50
9	80
10	40
11	60
12	10

ROWNUM, ROWID

ROWID : 데이터베이스에서 행의 주소를 반환합니다.

ROWNUM : 쿼리에 의해 반환되는 행의 번호를 출력합니다. (아주 중요)

```
SQL> SELECT ROWID, ROWNUM, employee_id, first_name  
2 FROM employees;
```

	ROWID	ROWNUM	EMPLOYEE_ID	FIRST_NAME
1	AAAEAbAAEAAAADNAAA	1	100	Steven
2	AAAEAbAAEAAAADNAAB	2	101	Neena
3	AAAEAbAAEAAAADNAAC	3	102	Lex
4	AAAEAbAAEAAAADNAAD	4	103	Alexander
5	AAAEAbAAEAAAADNAAE	5	104	Bruce
6	AAAEAbAAEAAAADNAAF	6	105	David
7	AAAEAbAAEAAAADNAAG	7	106	Valli
8	AAAEAbAAEAAAADNAAH	8	107	Diana
9	AAAEAbAAEAAAADNAAI	9	108	Nancy
10	AAAEAbAAEAAAADNAAJ	10	109	Daniel

...

ROWID를 분석해 보면 AAAEAbAAEAAAADNAAA에서...

- 처음 6자리 AAAEAb는 데이터 오브젝트 번호입니다.
- 다음 3자리 AAE는 상대적 파일 번호입니다.
- 세 번째 6자리 AAAADN은 블록의 번호입니다.
- 마지막 3자리 AAA는 블록내의 행 번호입니다.

SELECT문장 ROWNUM, ROWID 데이터 제한 데이터 정렬




질의에 의해 검색되는 행을 제한할 수 있습니다.

	FIRST_NAME	DEPARTMENT_ID
1	Steven	90
2	Neena	90
3	Lex	90
4	Alexander	60
5	Bruce	60
6	David	60
7	Valli	60
8	Diana	60
9	Nancy	100
10	Daniel	100
11	John	100
12	Ismael	100
13	Jose Manuel	100
14	Luis	100
15	Den	30
16	Alexander	30
17	Shelli	30

...

부서번호가 90인
모든 사원을 검색



	FIRST_NAME	DEPARTMENT_ID
1	Steven	90
2	Neena	90
3	Lex	90

WHERE 절을 사용하여 리턴되는 행을 제한합니다.

WHERE 절은 FROM 절 다음에 옵니다.

WHERE 절은 열 이름, 비교 연산자, 그리고 비교할 열 이름 또는 값의 목록으로 구성됩니다.

```
SELECT [DISTINCT] { *, column [[AS] alias],
... }
FROM table
[WHERE condition(s)];
```

구문형식에서...

- WHERE : 조건을 만족하는 행으로 질의를 제한합니다.
- condition(s) : 열 이름, 표현식, 상수 그리고 비교 연산자로 구성됩니다.

```
SQL> SELECT first_name, job_id, department_id
2 FROM employees
3 WHERE job_id='IT_PROG';
```

	⚡ FIRST_NAME	⚡ JOB_ID	⚡ DEPARTMENT_ID
1	Alexander	IT_PROG	60
2	Bruce	IT_PROG	60
3	David	IT_PROG	60
4	Valli	IT_PROG	60
5	Diana	IT_PROG	60

문자 스트링과 날짜 값은 단일 인용부호(' ')로 둘러싸여 있습니다.

문자 값은 대/소문자를 구분하고, 날짜 값은 날짜 형식을 구분합니다.

디폴트 날짜 형식은 'DD-MON-YY' 입니다.

```
SQL> SELECT first_name, last_name, hire_date
2 FROM employees
3 WHERE last_name='King';
```

	FIRST_NAME	LAST_NAME	HIRE_DATE
1	Janette	King	04/01/30
2	Steven	King	03/06/17

오라클은 날짜를 세기, 년, 월, 일, 시간, 분 그리고 초로 저장합니다.

디폴트 날짜 형식은 DD-MON-YY입니다. 위 결과에서 HIRE_DATE의 표시 형식이 04/01/30 인 이유는

SQL Developer의 환경설정에 데이터베이스 -> NLS -> 날짜 형식이 RR/MM/DD이기 때문입니다.



연산자	설명
=	같다
>	보다 크다
>=	보다 크거나 같다
<	보다 작다
<=	보다 작거나 같다
<>	같지 않다.

```
SQL> SELECT first_name, salary, hire_date
2 FROM employees
3 WHERE salary >= 15000;
```

	FIRST_NAME	SALARY	HIRE_DATE
1	Steven	24000	03/06/17
2	Neena	17000	05/09/21
3	Lex	17000	01/01/13

```
SQL> SELECT first_name, salary, hire_date
2 FROM employees
3 WHERE hire_date='04/01/30';
```

	FIRST_NAME	SALARY	HIRE_DATE
1	Janette	10000	04/01/30

```
SQL> SELECT first_name, salary, hire_date
2 FROM employees
3 WHERE first_name='Steven';
```

	FIRST_NAME	SALARY	HIRE_DATE
1	Steven	24000	03/06/17
2	Steven	2200	08/03/08

값의 범위에 해당하는 행을 출력하기 위해 BETWEEN 연산자를 사용합니다.

하한 값을 먼저 명시해야 합니다.

하한 값과 상한 값을 모두 포함합니다.

```
SQL> SELECT first_name, salary  
2 FROM employees  
3 WHERE salary BETWEEN 10000 AND 12000;
```

	FIRST_NAME	SALARY
1	Den	11000
2	Alberto	12000
3	Gerald	11000
4	Eleni	10500
5	Peter	10000
6	Janette	10000
7	Clara	10500
8	Lisa	11500
9	Harrison	10000
10	Ellen	11000
11	Hermann	10000

목록에 있는 값들과 비교하기 위해서 **IN 연산자**를 사용합니다.

```
SQL> SELECT employee_id, first_name, salary, manager_id
2 FROM employees
3 WHERE manager_id IN(101, 102, 103);
```

	EMPLOYEE_ID	FIRST_NAME	SALARY	MANAGER_ID
1	108	Nancy	12008	101
2	200	Jennifer	4400	101
3	203	Susan	6500	101
4	204	Hermann	10000	101
5	205	Shelley	12008	101
6	103	Alexander	9000	102
7	104	Bruce	6000	103
8	105	David	4800	103
9	106	Valli	4800	103
10	107	Diana	4200	103

```
SQL> SELECT first_name, last_name, job_id, department_id
2 FROM employees
3 WHERE job_id IN('IT_PROG', 'FI_MGR', 'AD_VP');
```

	FIRST_NAME	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	Neena	Kochhar	AD_VP	90
2	Lex	De Haan	AD_VP	90
3	Nancy	Greenberg	FI_MGR	100
4	Alexander	Hunold	IT_PROG	60
5	Bruce	Ernst	IT_PROG	60
6	David	Austin	IT_PROG	60
7	Valli	Pataballa	IT_PROG	60
8	Diana	Lorentz	IT_PROG	60

검색 스트링 값에 대한 와일드카드 검색을 위해서 **LIKE** 연산자를 사용합니다.

검색 조건은 리터럴 문자나 숫자를 포함할 수 있습니다.

%(percent)는 **문자가 없거나 또는 하나 이상을** 나타냅니다.

_(under score)는 **하나의 문자를** 나타냅니다..

```
SQL> SELECT first_name, last_name, job_id, department_id
2  FROM   employees
3  WHERE  job_id LIKE 'IT%';
```

	⚡ FIRST_NAME	⚡ LAST_NAME	⚡ JOB_ID	⚡ DEPARTMENT_ID
1	Alexander	Hunold	IT_PROG	60
2	Bruce	Ernst	IT_PROG	60
3	David	Austin	IT_PROG	60
4	Valli	Pataballa	IT_PROG	60
5	Diana	Lorentz	IT_PROG	60

LIKE 연산자는 BETWEEN 비교에 대한 단축키로 사용될 수 있습니다. 다음 예는, 2003년 1월 1일과 12월 31일 사이에 입사한 모든 사원의 이름과 입사일을 출력 합니다.

```
SQL> SELECT first_name, hire_date
2   FROM   employees
3   WHERE  hire_date LIKE '03%';
```

	FIRST_NAME	HIRE_DATE
1	Steven	03/06/17
2	Alexander	03/05/18
3	Payam	03/05/01
4	Renske	03/07/14
5	Trenna	03/10/17
6	Jennifer	03/09/17

“%” 와 “_” 기호는 리터럴 문자의 결합하여 사용될 수 있습니다. 위의 예는 이메일의 두 번째 문자가 “A”인 모든 사원의 이름과 이메일을 출력 합니다.

```
SQL> SELECT first_name, email
2   FROM   employees
3   WHERE  email LIKE '_A%';
```

	FIRST_NAME	EMAIL
1	Ellen	EABEL
2	Sundar	SANDE
3	Mozhe	MATKINSO
4	David	DAUSTIN
5	James	JAMRLOW

IS NULL 연산자로 Null 값을 테스트 합니다.

IS NULL 연산자는 Null인 값에 대해서 테스트 합니다. Null 값은 값이 없거나, 알 수 없거나, 또는 적용할 수 없음을 의미합니다. 그러므로 Null 값은 어떤 값과 같거나 또는 다를 수 없으므로 (=)로는 테스트 할 수가 없습니다.

```
SQL> SELECT first_name, manager_id
2 FROM employees
3 WHERE manager_id IS NULL;
```

	FIRST_NAME	MANAGER_ID	JOB_ID
1	Steven	(null)	AD_PRES

보너스(COMMISSION_PCT)가 없는 모든 사원에 대해서 이름과 직무 그리고 보너스를 출력하기 위해서 다음의 SQL 문장을 사용합니다.

```
SQL> SELECT first_name, job_id, commission_pct
2 FROM employees
3 WHERE commission_pct IS NULL;
```

NULL이 아닌 컬럼을 검색할 경우에는 IS NOT NULL을 이용합니다.

```
SQL> SELECT first_name, job_id, commission_pct
2 FROM employees
3 WHERE commission_pct IS NOT NULL;
```

스크립트 출력 x | 결과의 결과 x

SQL | 인출된 모든 행: 72(0.004초)

	FIRST_NAME	JOB_ID	COMMISSION_PCT
1	Steven	AD_PRES	(null)
2	Neena	AD_VP	(null)
3	Lex	AD_VP	(null)
4	Alexander	IT_PROG	(null)
5	Bruce	IT_PROG	(null)
6	David	IT_PROG	(null)

스크립트 출력 x | 결과의 결과 x

SQL | 인출된 모든 행: 35(0.001초)

	FIRST_NAME	JOB_ID	COMMISSION_PCT
1	John	SA_MAN	0.4
2	Karen	SA_MAN	0.3
3	Alberto	SA_MAN	0.3
4	Gerald	SA_MAN	0.3
5	Eleni	SA_MAN	0.2
6	Paul	SA_MAN	0.4

AND는 양쪽의 조건이 참이어야 TRUE를 리턴합니다.

OR는 한쪽의 조건이 참이면 TRUE를 리턴합니다.

NOT 연산자는 뒤의 조건에 반대되는 결과를 리턴합니다.

```
SQL> SELECT first_name, job_id, salary
2 FROM employees
3 WHERE job_id='IT_PROG' AND salary>=5000;
```

	FIRST_NAME	JOB_ID	SALARY
1	Alexander	IT_PROG	9000
2	Bruce	IT_PROG	6000

```
SQL> SELECT first_name, job_id, salary
2 FROM employees
3 WHERE job_id='IT_PROG' OR salary>=5000;
```

SQL | 인출된 모든 행: 61(0.006초)

	FIRST_NAME	JOB_ID	SALARY
1	Steven	AD_PRES	24000
2	Neena	AD_VP	17000
3	Lex	AD_VP	17000
4	Alexander	IT_PROG	9000
5	Bruce	IT_PROG	6000
6	David	IT_PROG	4800
7	Valli	IT_PROG	4800
8	Diana	IT_PROG	4200
9	Nancy	FI_MGR	12008
10	Daniel	FI_ACCOUNT	9000

- AND, OR, NOT 연산자의 우선순위는 다른 모든 비교연산자에 비하여 낮습니다.
- OR 연산자보다 AND 연산자가 우선순위가 높습니다.

우선순위	연산자
1	모든 비교 연산자
2	NOT
3	AND
4	OR


```
SQL> SELECT first_name, job_id, salary
2 FROM employees
3 WHERE job_id='IT_PROG'
4 OR job_id='FI_MGR'
5 AND salary >= 6000;
```

	⚡ FIRST_NAME	⚡ JOB_ID	⚡ SALARY
1	Alexander	IT_PROG	9000
2	Bruce	IT_PROG	6000
3	David	IT_PROG	4800
4	Valli	IT_PROG	4800
5	Diana	IT_PROG	4200
6	Nancy	FI_MGR	12008

위의 예에는 두 개의 조건이 있습니다.

- 첫 번째 조건은 잡 아이디가 IT_PROG입니다.
- 두 번째 조건은 잡 아이디가 FI_MGR 이고 급여가 \$6000 이상 입니다.

```
SQL> SELECT first_name, job_id, salary
2 FROM employees
3 WHERE (job_id='IT_PROG' OR job_id='FI_MGR')
4 AND salary >= 6000;
```

	⚡ FIRST_NAME	⚡ JOB_ID	⚡ SALARY
1	Nancy	FI_MGR	12008
2	Alexander	IT_PROG	9000
3	Bruce	IT_PROG	6000

위의 예에는 두 개의 조건이 있습니다.

- 첫 번째 조건은 잡 아이디가 IT_PROG 이거나 FI_MGR입니다.
- 두 번째 조건은 급여가 \$6000 이상입니다.

SELECT문장 ROWNUM, ROWID 데이터 제한 데이터 정렬



질의에 의해 검색되는 행을 정렬할 수 있습니다.

ORDER BY 절은 **SELECT 문장의 가장 뒤에 옵니다.**

ASC : 오름차순으로 정렬합니다. 기본값입니다.

DESC : 내림차순으로 정렬합니다.

```
SELECT  expr
FROM    table
[WHERE  condition(s)]
[ORDER BY {column|expr, ...} [[ASC||DESC]]];
```

```
SQL> SELECT  first_name, hire_date
2  FROM      employees
3  ORDER BY  hire_date;
```

	FIRST_NAME	HIRE_DATE
1	Lex	01/01/13
2	William	02/06/07
3	Hermann	02/06/07
4	Susan	02/06/07
5	Shelley	02/06/07
6	Daniel	02/08/16
7	Nancy	02/08/17
8	Den	02/12/07
9	Payam	03/05/01
10	Alexander	03/05/18

...

ORDER BY 절에서 열 이름 뒤에 DESC 키워드를 명시합니다.

```
SQL> SELECT  first_name, hire_date  
2  FROM      employees  
3  ORDER BY  hire_date DESC;
```

	FIRST_NAME	HIRE_DATE
1	Sundita	08/04/21
2	Amit	08/04/21
3	Sundar	08/03/24
4	Steven	08/03/08
5	David	08/02/23
6	Hazel	08/02/06
7	Girard	08/02/03
8	Eleni	08/01/29
9	Mattea	08/01/24
10	Douglas	08/01/13

...

ORDER BY 절에 열 별칭을 사용할 수 있습니다.

```
SQL> SELECT first_name, salary*12 AS annsal  
2 FROM employees  
3 ORDER BY annsal;
```

	FIRST_NAME	ANNSAL
1	TJ	25200
2	Steven	26400
3	Hazel	26400
4	Ki	28800
5	James	28800
6	Karen	30000
7	James	30000
8	Joshua	30000
9	Peter	30000
10	Martha	30000

...

1. 모든 사원의 사원번호, 이름, 입사일, 급여를 출력하세요.
2. 모든 사원의 이름과 성을 붙여 출력하세요. 열 별칭은 name으로 하세요.
3. 50번 부서 사원의 모든 정보를 출력하세요.
4. 50번 부서 사원의 이름, 부서번호, 직무아이디를 출력하세요.
5. 모든 사원의 이름, 급여 그리고 300달러 인상된 급여를 출력하세요.
6. 급여가 10000보다 큰 사원의 이름과 급여를 출력하세요.
7. 보너스를 받는 사원의 이름과 직무, 보너스율을 출력하세요.
8. 2003년도 입사한 사원의 이름과 입사일 그리고 급여를 출력하세요.(BETWEEN 연산자 사용)
9. 2003년도 입사한 사원의 이름과 입사일 그리고 급여를 출력하세요.(LIKE 연산자 사용)
10. 모든 사원의 이름과 급여를 급여가 많은 사원부터 적은 사원순서로 출력하세요.
11. 위 질의를 60번 부서의 사원에 대해서만 질의하세요. (컬럼: department_id)
12. 직무아이디가 IT_PROG 이거나, SA_MAN인 사원의 이름과 직무아이디를 출력하세요.
13. Steven King 사원의 정보를 "Steven King 사원의 급여는 24000달러 입니다" 형식으로 출력하세요.
14. 매니저(MAN) 직무에 해당하는 사원의 이름과 직무아이디를 출력하세요. (컬럼:job_id)
15. 매니저(MAN) 직무에 해당하는 사원의 이름과 직무아이디를 직무아이디 순서대로 출력하세요.

- 제한시간 : 30분
- EMPLOYEES 테이블 데이터를 출력해야 합니다.