

# Select

강사 : 강병준

# SELECT

## SELECT

{ } 중에서 선택

```
SELECT      [DISTINCT] {*,column [Alias],...}  
FROM        테이블명 ;  
[WHERE      condition]  
[ORDER BY   {column, expression} [ASC | DESC]];
```

반드시 사용  
또는 '/'

SELECT	: 원하는 컬럼을 선택
*	: 테이블의 모든 column 출력
alias	: 해당 column에 대한 다른 이름 부여
DISTINCT	: 중복 행 제거 옵션
FROM	: 원하는 데이터가 저장된 테이블 명을 기술.
WHERE	: 조회되는 행을 제한(선택)
condition	: column, 표현식, 상수 및 비교 연산자
ORDER BY	: 정렬을 위한 옵션(ASC:오름차순(Default),DESC내림차순)

# SELECT

## SQL 문장 작성법

1. SQL 문장은 대소문자를 구별하지 않습니다.
2. SQL 문장은 한 줄 또는 여러 줄에 입력될 수 있습니다.
3. 하나의 명령어는 여러 줄에 나누거나 단축될 수 없습니다.
4. 절은 보통 읽고 편집하기 쉽게 줄을 나누도록 합니다.(권장)
5. 탭과 줄 넣기(들여쓰기)는 코드를 보다 읽기 쉽게 하기 위해 사용됩니다.(권장)
6. 일반적으로 키워드는 대문자로 입력합니다.
7. 키워드를 제외한 다른 모든 단어, 즉 테이블 이름, 열 이름은 소문자로 입력합니다.(권장)
8. SQL\*Plus에서 SQL 문장은 SQL 프롬프트에 입력되며 1라인 이후의 라인은 라인 번호가 붙습니다. 가장 최근의 명령어가 1개가 SQL buffer에 저장됩니다.

## SQL 문장 실행

1. 마지막 절의 끝에 ";"를 기술하여 명령의 끝을 표시
2. 버퍼에서 마지막 라인에 슬래시를 넣습니다.(OS의 Editor사용시)
3. SQL프롬프트에 슬래시를 입력합니다.(SQL Buffer의 내용 실행)
4. SQL프롬프트에서 SQL\*Plus RUN 명령어를 실행합니다. (SQL Buffer의 내용 실행)

# SELECT

```
SQL> SELECT * FROM TAB;
```

TNAME	TABTYPE	CLUSTERID
BONUS	TABLE	
DEPT	TABLE	
EMP	TABLE	
SALGRADE	TABLE	

```
SQL>
```

출력 결과를 보면 SCOTT 계정이 사용할 수 있는 객체는 BONUS, DEPT, EMP, SALGRADE이며 모두 TABLE임을 알 수 있다. 그렇다면 DEPT 테이블에는 어떤 컬럼이 있는지 확인해보자.

```
SQL> DESC DEPT
```

이름	널?	유형
DEPTNO	NOT NULL	NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

# SELECT

## 1. 모든 열 선택

SELECT 키워드에 "\*" 을 사용하여 테이블의 열 데이터 모두를 조회할 수 있습니다.

## 2. SCOTT이 소유하고 있는 EMP Table의 모든 데이터를 출력

```
SELECT * FROM emp;
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	80/12/17	800	(null)	20
2	7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
3	7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
4	7566	JONES	MANAGER	7839	81/04/02	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
6	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
7	7782	CLARK	MANAGER	7839	81/06/09	2450	(null)	10
8	7788	SCOTT	ANALYST	7566	87/04/19	3000	(null)	20
9	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
10	7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
11	7876	ADAMS	CLERK	7788	87/05/23	1100	(null)	20
12	7900	JAMES	CLERK	7698	81/12/03	950	(null)	30
13	7902	FORD	ANALYST	7566	81/12/03	3000	(null)	20
14	7934	MILLER	CLERK	7782	82/01/23	1300	(null)	10

# SELECT

## 1. 특정 Column 선택

1) 테이블의 특정 Column을 검색하고자 할 경우 Column이름을 ","로 구분하여 명시함으로써 특정 Column을 출력할 수 있습니다.

2) 출력 순서는 SELECT문 뒤에 기술한 Column의 순서대로 출력됩니다.

## 2. SCOTT이 소유하고 있는 EMP Table에서 empno,ename,sal,job 를 출력

```
SELECT empno,ename,sal,job FROM emp;
```

	EMPNO	ENAME	SAL	JOB
1	7369	SMITH	800	CLERK
2	7499	ALLEN	1600	SALESMAN
3	7521	WARD	1250	SALESMAN
4	7566	JONES	2975	MANAGER
5	7654	MARTIN	1250	SALESMAN
6	7698	BLAKE	2850	MANAGER
7	7782	CLARK	2450	MANAGER
8	7788	SCOTT	3000	ANALYST
9	7839	KING	5000	PRESIDENT
10	7844	TURNER	1500	SALESMAN
11	7876	ADAMS	1100	CLERK
12	7900	JAMES	950	CLERK
13	7902	FORD	3000	ANALYST
14	7934	MILLER	1300	CLERK

# WHERE 조건과 비교 연산자

- 원하는 로우만 얻으려면 다음과 같이 로우를 제한하는 조건을 SELECT 문에 WHERE 절을 추가하여 제시해야 합니다.

형식	SELECT * [column1, column2, .. ,columnn] FROM table_name <b>WHERE</b> <u>조건절</u> ;
----	--

- 조건 절은 다음의 세 부분으로 구성이 됩니다.

조건절의 구성	WHERE SAL >= 3000; ①컬럼 ②연산자 ③비교대상값
---------	---------------------------------------

# 비교 연산자

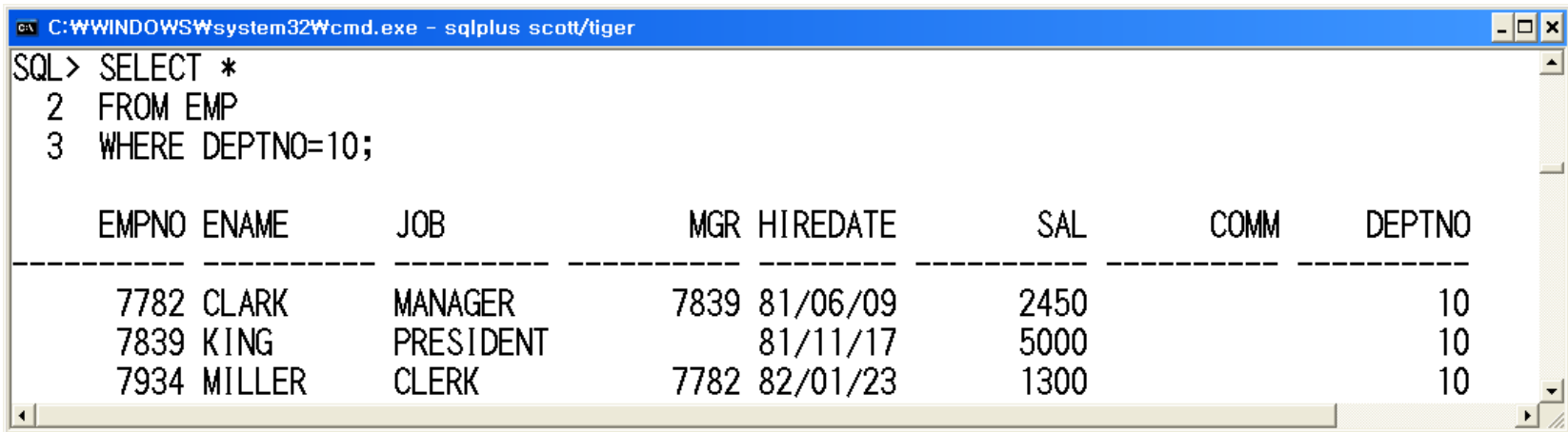
연산자	의 미	예 제
=	같다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL=3000;
>	보다 크다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL>3000;
<	보다 작다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<3000;
>=	보다 크거나 같다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL>=3000;
<=	보다 작거나 같다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<=3000;
<>, !=, ^=	다르다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<>3000;



# 비교 연산자

예

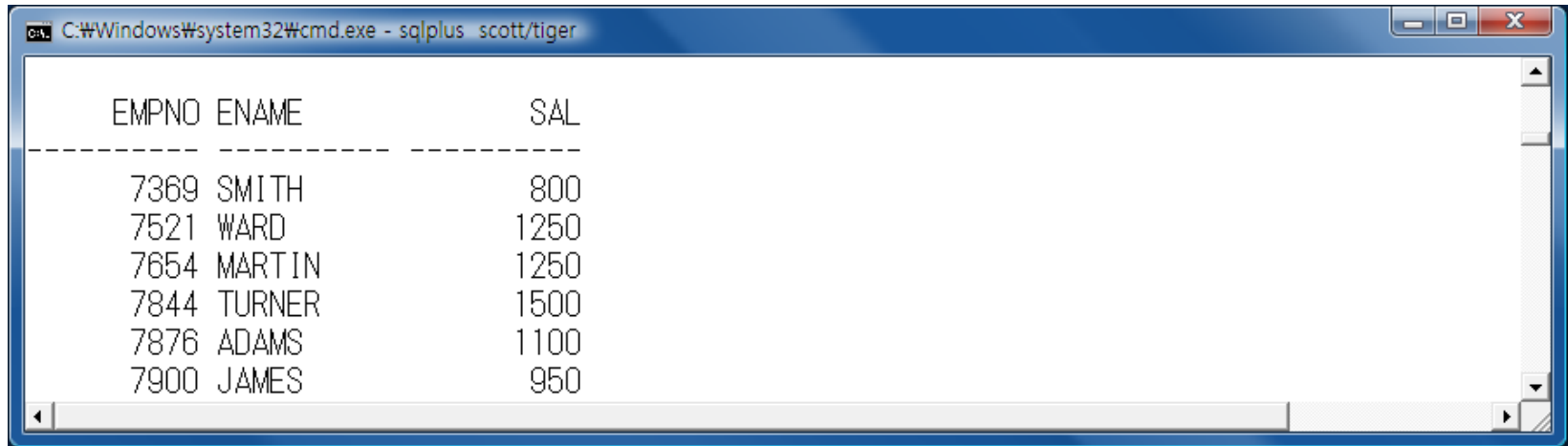
```
SELECT *  
FROM EMP  
WHERE DEPTNO=10;
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7839	KING	PRESIDENT		81/11/17	5000		10
7934	MILLER	CLERK	7782	82/01/23	1300		10

# 비교 연산자

1. 급여가 1500 이하인 사원의 사원번호, 사원 이름, 급여를 출력하는 SQL 문을 작성해 보시오.



EMPNO	ENAME	SAL
7369	SMITH	800
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950

- <힌트> 사원 정보가 저장된 테이블의 이름은 EMP이고, 사원번호 컬럼은 EMPNO, 사원이름 컬럼은 ENAME, 급여 컬럼은 SAL입니다.

# LITERAL 문자 STRING

1. LITERAL은 열 이름이나 열 별칭이 아닌 SELECT목록에 포함되어 있는 문자, 표현식, 숫자
2. RETURN되는 각각의 행에 대해 출력됩니다.
3. LITERAL과 STRING은 질의 결과에 포함될 수 있으며 SELECT목록에서 열과 똑같이 취급됩니다.
4. 날짜와 문자 LITERAL은 단일 인용 부호(' ')를 사용하여야 하고 숫자 LITERAL은 사용하지 않습니다.
5. SELECT절에 포함된 LITERAL은 문자, 표현식, 숫자입니다.
6. 날짜와 문자 LITERAL 값은 단일 인용부호(' ') 안에 있어야 합니다.
7. 각각의 문자 STRING은 RETURN된 각 행에 대한 결과입니다.

# 문자 데이터 조회

1. 이전 예제에서 비교 연산자를 하기 위해서 다루었던 컬럼들은 수치 형태로 선언되었습니다. 이번에는 급여가 아닌 사원 이름 같은 문자 데이터를 조회해 봅시다.
2. 다음은 이름이 FORD인 사원의 사원번호(EMPNO)와 사원이름(ENAME)과 급여(SAL)을 출력하는 예제입니다.

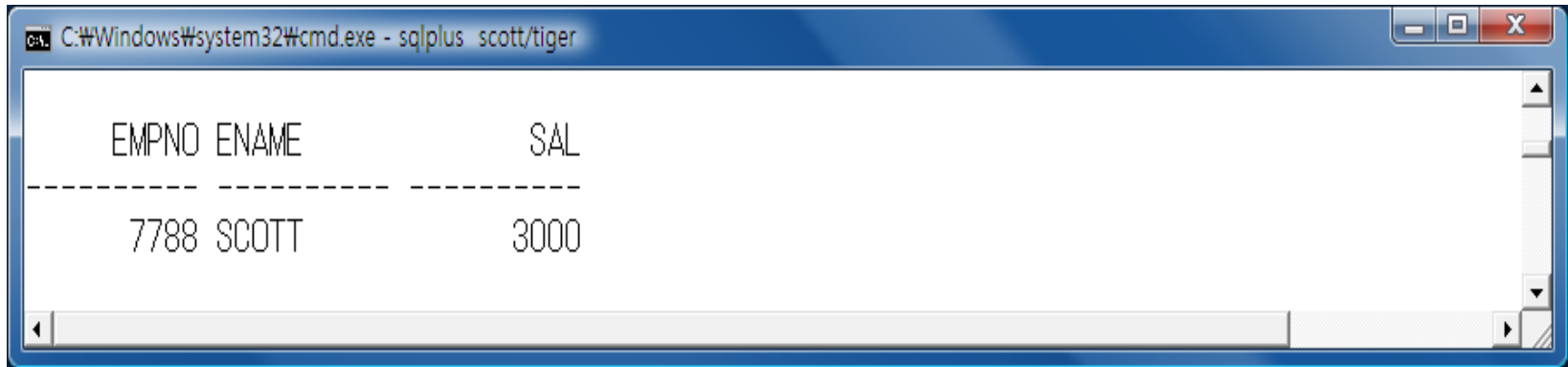
예

```
SELECT EMPNO, ENAME, SAL  
FROM EMP  
WHERE ENAME='FORD';
```

3. SQL에서 문자열이나 날짜는 반드시 단일 따옴표(single quotation) 안에 표시해야 합니다.
4. SQL문에 사용되는 키워드인 SELECT 나 FROM 이나 WHERE 등은 대소문자를 구별하지 않지만 테이블 내에 저장된 데이터 값은 대소문자를 구분하기에 WHERE ENAME='ford'와 같이 기술하면 사원이름이 FORD 인 사원을 찾을 수 없습니다.

# 문자 데이터 조회

- 사원 이름이 SCOTT 인 사원의 사원번호, 사원 이름, 급여를 출력하는 SQL 문을 작성해 보시오.



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The window displays the output of an SQL query, which is a table with three columns: EMPNO, ENAME, and SAL. The table has one data row showing the employee with EMPNO 7788, ENAME SCOTT, and SAL 3000.

EMPNO	ENAME	SAL
7788	SCOTT	3000

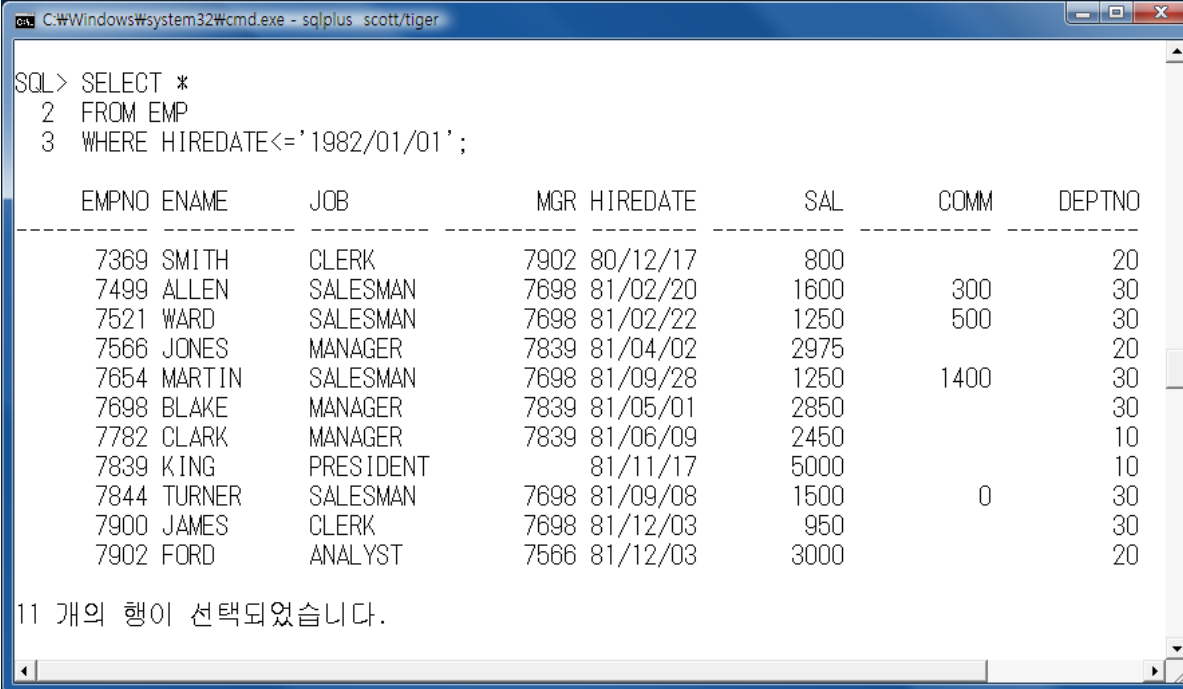
- <힌트> 사원 정보가 저장된 테이블의 이름은 EMP이고, 사원번호 컬럼은 EMPNO, 사원이름 컬럼은 ENAME, 급여 컬럼은 SAL입니다.

# 날짜 데이터 조회

- 82년 1월 1일 이후에 입사한 사원을 조회하려면 어떻게 해야 할까요?  
날짜는 문자열과 마찬가지로 단일 따옴표 안에 기술해야 합니다.
- 다음은 82년 1월 1일 이후에 입사한 사원을 출력하는 예제입니다.

예

```
SELECT *  
FROM EMP  
WHERE HIREDATE>='82/01/01';
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80/12/17	800		20
7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
7566	JONES	MANAGER	7839	81/04/02	2975		20
7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7839	KING	PRESIDENT		81/11/17	5000		10
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
7900	JAMES	CLERK	7698	81/12/03	950		30
7902	FORD	ANALYST	7566	81/12/03	3000		20

11 개의 행이 선택되었습니다.

# 논리 연산자

- 오라클에서 사용 가능한 논리 연산자 AND나 OR나 NOT가 있습니다

연산자	의미
AND	두 가지 조건을 모두 만족해야만 검색할 수 있다. <code>SELECT * FROM emp WHERE deptno=10 AND job='MANAGER';</code>
OR	두 가지 조건 중에서 한 가지만 만족하더라도 검색할 수 있다. <code>SELECT * FROM emp WHERE deptno=10 OR job='MANAGER';</code>
NOT	조건에 만족하지 못하는 것만 검색한다. <code>SELECT * FROM emp WHERE NOT deptno=10;</code>

# AND 연산자

1. 두 가지 조건을 모두 만족할 경우에만 검색할 수 있도록 하기 위해서는 AND 연산자를 사용
2. 다음은 AND 연산자가 조건에 따라 어떤 결과가 출력되는지를 나타내는 표

두 조건이 모두 만족할 경우에만 결과가 참이고, 조건 중 하나라도 만족하지 않으면 결과가 거짓

조건 1	조건2	and
참	참	참
참	거짓	거짓
거짓	참	거짓
거짓	거짓	거짓



# AND 연산자

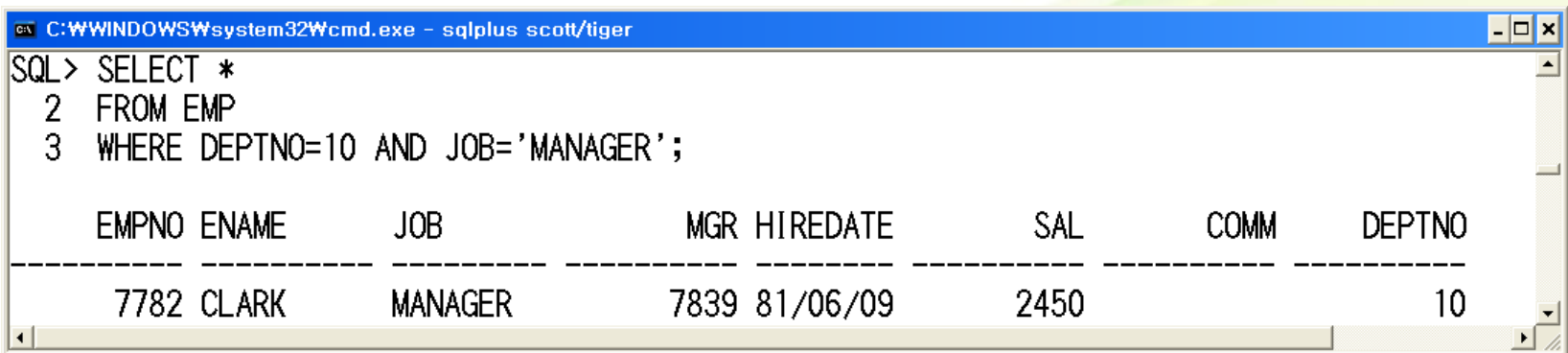
1. 10번 부서 소속인 사원들 중에서 직급이 MANAGER인 사람을 검색하여 사원명, 부서번호, 직급을 출력하려고 한다면 두 가지 조건을 제시해야 함

[조건1] 10번 부서 소속인 사원 : DEPTNO=10

[조건2] 직급이 MANAGER인 사원 : JOB='MANAGER'

예

```
SELECT *  
FROM EMP  
WHERE DEPTNO=10 AND JOB='MANAGER';
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The prompt is "SQL>". The user has entered the following SQL query:

```
SQL> SELECT *  
2 FROM EMP  
3 WHERE DEPTNO=10 AND JOB='MANAGER';
```

The query returns one row of data. The columns are EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. The data row shows EMPNO 7782, ENAME CLARK, JOB MANAGER, MGR 7839, HIREDATE 81/06/09, SAL 2450, COMM, and DEPTNO 10.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	81/06/09	2450		10

# OR 연산자

1. 두 가지 조건 중에서 한 가지만 만족하더라도 검색할 수 있도록 하기 위해서는 OR 연산자를 사용
2. OR 연산자가 조건에 따라 어떤 결과가 출력되는지를 나타내는 표

**OR** 연산자는 두 조건에 모두 만족하지 않을 경우는 결과가 거짓이고 제시한 조건에 한 가지라도 만족하면 결과가 참

조건 1	조건2	or
참	참	참
참	거짓	참
거짓	참	참
거짓	거짓	거짓

# OR 연산자

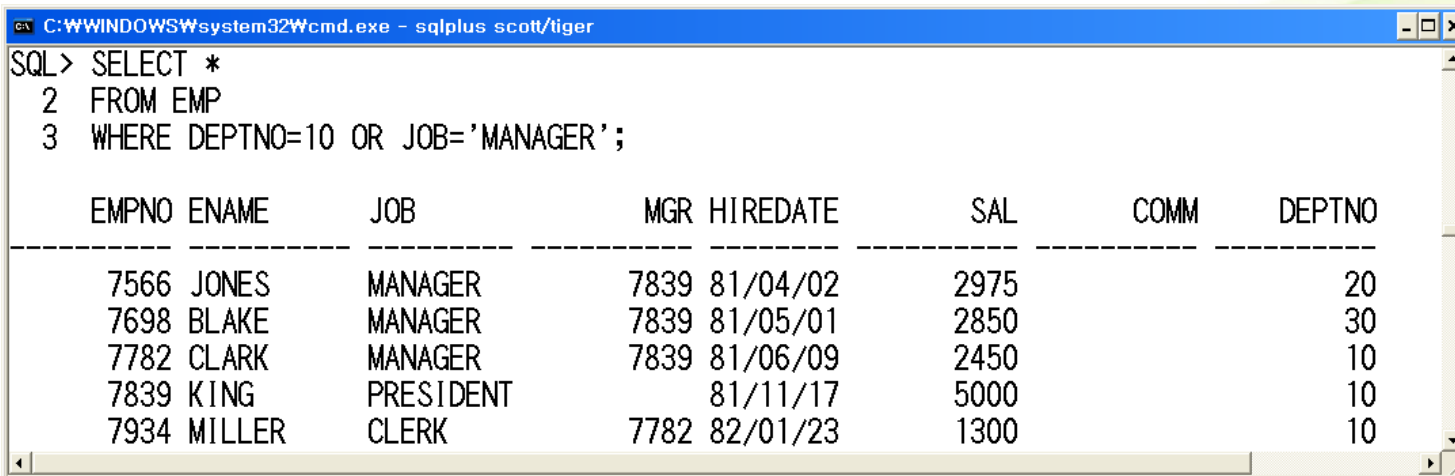
10번 부서에 소속된 사원이거나 직급이 MANAGER인 사람을 검색하여  
사원명, 부서번호, 직급을 출력.

[조건1] 10번 부서 소속인 사원 : DEPTNO=10

[조건2] 직급이 MANAGER인 사원 : JOB='MANAGER'

예

```
SELECT *  
FROM EMP  
WHERE DEPTNO=10 OR JOB='MANAGER';
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe - sqlplus scott/tiger". The user has entered the following SQL query:

```
SQL> SELECT *  
2 FROM EMP  
3 WHERE DEPTNO=10 OR JOB='MANAGER';
```

The query results are displayed in a table with the following columns: EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. The results show five rows of data:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	81/04/02	2975		20
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7839	KING	PRESIDENT		81/11/17	5000		10
7934	MILLER	CLERK	7782	82/01/23	1300		10

# NOT 연산자

1. 10번 부서에 소속된 직원만 제외하고 나머지 직원의 정보를 출력하려면 어떻게 해야 할까요? 이러한 조건을 제시하기 위해서 사용하는 논리 연산자가 바로 NOT
2. NOT 연산자는 참은 거짓으로 거짓은 참으로 즉 반대되는 논리값을 구하는 연산자

다음은 **NOT** 연산자가 논리값에 의해서 어떤 결과가 출력되는지를 나타내는 표

조건	NOT
참	거짓
거짓	참

# NOT 연산자

1. 이 조건 앞에 NOT을 붙이면 부서번호가 10번이 아닌 사원들에 대해서만 검색
2. 다음은 부서번호가 10번이 아닌 사원의 사원이름, 부서번호, 직급을 출력

예

```
SELECT *  
FROM EMP  
WHERE NOT DEPTNO=10;
```

예

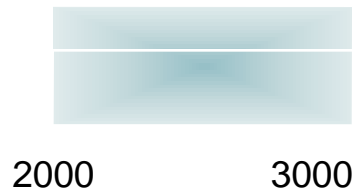
```
SELECT *  
FROM EMP  
WHERE DEPTNO<>10;
```

# 논리 연산자의 다양한 활용

## 2000에서 3000 사이의 급여를 받는 사원을 조회

[조건1] 급여가 2000 이상 :  $sal \geq 2000$

[조건2] 급여가 3000 이하 :  $sal \leq 3000$



예

```
SELECT *  
FROM EMP  
WHERE SAL >= 2000 AND SAL <= 3000;
```

# 논리 연산자

1. 논리 연산자: AND, OR, NOT
2. EMP 테이블에서 sal이 1100이상이고 job이 MANAGER 인 사원의 empno,ename,job,sal,hiredate,deptno를 출력

```
SELECT empno,ename,job,sal,hiredate,deptno FROM emp  
WHERE sal >= 1100 AND job = 'MANAGER'
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA O

Results:

	EMPNO	ENAME	JOB	SAL	HIREDATE	DEPTNO
1	7566	JONES	MANAGER	2975	81/04/02	20
2	7698	BLAKE	MANAGER	2850	81/05/01	30
3	7782	CLARK	MANAGER	2450	81/06/09	10

# 논리 연산자

1. EMP 테이블에서 sal이 1100이상이거나 job이 MANAGER 인 사원의 empno,ename,job,sal,hiredate,deptno를 출력

```
SELECT empno,ename,job,sal,hiredate,deptno FROM emp  
WHERE sal >= 1100 OR job = 'MANAGER'
```

Results:

	EMPNO	ENAME	JOB	SAL	HIREDATE	DEPTNO
1	7499	ALLEN	SALESMAN	1600	81/02/20	30
2	7521	WARD	SALESMAN	1250	81/02/22	30
3	7566	JONES	MANAGER	2975	81/04/02	20
4	7654	MARTIN	SALESMAN	1250	81/09/28	30
5	7698	BLAKE	MANAGER	2850	81/05/01	30
6	7782	CLARK	MANAGER	2450	81/06/09	10
7	7788	SCOTT	ANALYST	3000	87/04/19	20
8	7839	KING	PRESIDENT	5000	81/11/17	10
9	7844	TURNER	SALESMAN	1500	81/09/08	30
10	7876	ADAMS	CLERK	1100	87/05/23	20
11	7902	FORD	ANALYST	3000	81/12/03	20
12	7934	MILLER	CLERK	1300	82/01/23	10



# 논리 연산자

1. NOT 연산자는 BETWEEN, LIKE, IS NULL과 같은 다른 SQL 연산자와 함께 사용
2. EMP 테이블에서 job이 MANAGER, CLERK, ANALYST가 아닌 사원의 empno, ename, job, sal, deptno를 출력

```
SELECT empno, ename, job, sal, deptno FROM emp  
WHERE job NOT IN ('MANAGER', 'CLERK', 'ANALYST')
```

Results:

	EMPNO	ENAME	JOB	SAL	DEPTNO
1	7499	ALLEN	SALESMAN	1600	30
2	7521	WARD	SALESMAN	1250	30
3	7654	MARTIN	SALESMAN	1250	30
4	7839	KING	PRESIDENT	5000	10
5	7844	TURNER	SALESMAN	1500	30

# 논리 연산자

1. 우선 순위 규칙: 괄호 > 모든 비교 연산자 > NOT > AND > OR
2. emp 테이블에서 job이 SALESMAN 이거나 PRESIDENT이고 sal이 1500이 넘는 사원의 empno,ename,job,sal를 출력

```
SELECT empno,ename,job,sal FROM emp  
WHERE job = 'SALESMAN' OR job = 'PRESIDENT' AND sal > 1500
```

Results:

	EMPNO	ENAME	JOB	SAL
1	7499	ALLEN	SALESMAN	1600
2	7521	WARD	SALESMAN	1250
3	7654	MARTIN	SALESMAN	1250
4	7839	KING	PRESIDENT	5000
5	7844	TURNER	SALESMAN	1500

# 논리 연산자

1. emp 테이블에서 (job이 SALESMAN 이거나 PRESIDENT)이고 sal이 1500이 넘는 사원의 empno,ename,job,sal를 출력

```
SELECT empno,ename,job,sal FROM emp  
WHERE (job = 'SALESMAN' OR job = 'PRESIDENT') AND sal > 1500
```

Results:

	EMPNO	ENAME	JOB	SAL
1	7499	ALLEN	SALESMAN	1600
2	7839	KING	PRESIDENT	5000

# 논리 연산자

**커미션이 300 이거나 500 이거나 1400 인 사원을 검색**

[조건1] 커미션이 300 : COMM=300

[조건2] 커미션이 500 : COMM=500

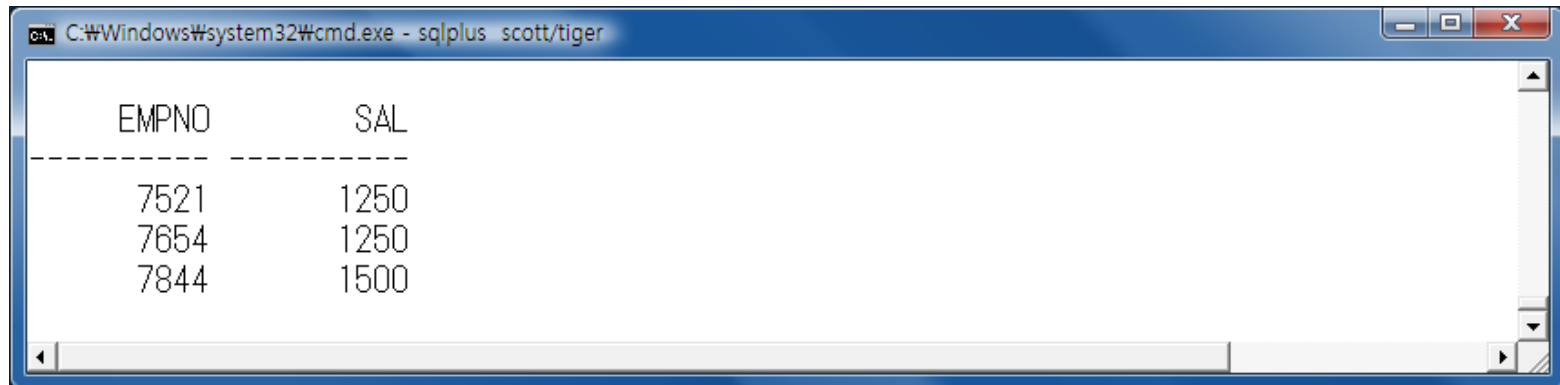
[조건3] 커미션이 1400 : COMM=1400

예

```
SELECT *  
FROM EMP  
WHERE COMM=300 OR COMM=500 OR COMM=1400;
```

# 논리 연산자

부서번호는 DEPTNO로, 부서명은 DNAME으로 정해져 있으므로 다음과 같은 쿼리문의 결과는 왼쪽 그림과 같이 컬럼 헤딩에도 역시 DEPTNO, DNAME으로 출력됩니다. 3. 7521 이거나 7654 이거나 7844 인 직원들의 직원 번호와 급여를 검색하는 쿼리문을 비교 연산자와 OR 논리 연산자 사용하여 작성하시오.



EMPNO	SAL
7521	1250
7654	1250
7844	1500

# 산술 표현식

1. 데이터가 출력 되는 방식을 수정하거나 계산을 수행하고자 할 때 산술 표현식
2. 산술 표현식은 열 이름, 숫자 상수, 문자 상수, 산술 연산자를 포함할 수 있으며 연산자는 +(Add), -(Subtract), \*(Multiply), /(Divide)을 사용
3. SELECT문장에서는 FROM절을 제외한 SQL문장의 절에서 사용할 수 있습니다.
4. 산술 표현식이 하나 이상의 연산자를 포함한다면 일반적인 산술 연산자 우선 순위를 적용
5. emp 테이블의 Salary를 300증가 시키기 위해 덧셈 연산자를 사용하고  
ename, sal, sal+300을 출력

SQL>SELECT ename, sal, sal+300 FROM emp;

	ENAME	SAL	SAL+300
1	SMITH	800	1100
2	ALLEN	1600	1900
3	WARD	1250	1550
4	JONES	2975	3275
5	MARTIN	1250	1550
6	BLAKE	2850	3150
7	CLARK	2450	2750
8	SCOTT	3000	3300
9	KING	5000	5300
10	TURNER	1500	1800
11	ADAMS	1100	1400
12	JAMES	950	1250
13	FORD	3000	3300
14	MILLER	1300	1600

# 산술 표현식

## 주의

1. 계산된 결과 열 SAL+300은 EMP테이블의 새로운 열이 아니고 단지 디스플레이를 위한 것
2. 디폴트로 새로운 열의 이름 sal+300은 생성된 계산식으로부터 유래
3. SQL\*Plus는 산술 연산자 앞뒤의 공백을 무시

## null값의 처리

1. 행이 특정 열에 대한 데이터 값이 없다면 값은 null
2. null값은 이용할 수 없거나 지정되지 않았거나, 알 수 없거나 또는 적용할 수 없는 값
3. null값은 0이나 공백과는 다르며 0은 숫자이며 공백은 문자
4. 열이 NOT NULL로 정의되지 않았거나 열이 생성될 때 PRIMARY KEY로 정의되지 않았다면 열은 null값을 포함
5. EMP 테이블의 COMM열에서 오직 SALESMAN만이 보너스를 받을 수 있음
6. 널 값을 포함한 산술 표현식 결과는 NULL
7. column에 데이터 값이 없으면 그 값 자체가 널 또는 널 값을 포함
8. 널 값은 1바이트의 내부 저장 장치를 오버헤드로 사용하고 있으며 어떠한 datatype column들이라도 널 값을 포함

# 산술 표현식

emp 테이블에서 empno, ename, sal, comm, sal+comm/100을 출력

```
SELECT empno,ename,sal,comm,sal+comm/100 FROM emp
```

	EMPNO	ENAME	SAL	COMM	SAL+COMM/100
1	7369	SMITH	800	(null)	(null)
2	7499	ALLEN	1600	300	1603
3	7521	WARD	1250	500	1255
4	7566	JONES	2975	(null)	(null)
5	7654	MARTIN	1250	1400	1264
6	7698	BLAKE	2850	(null)	(null)
7	7782	CLARK	2450	(null)	(null)
8	7788	SCOTT	3000	(null)	(null)
9	7839	KING	5000	(null)	(null)
10	7844	TURNER	1500	0	1500
11	7876	ADAMS	1100	(null)	(null)
12	7900	JAMES	950	(null)	(null)
13	7902	FORD	3000	(null)	(null)
14	7934	MILLER	1300	(null)	(null)



# NULL 값의 정의 및 처리

NULL이란 이용 불가능한(Unavailable), 지정되지 않은(Unassigned), 알 수 없는(Unknown), 적용할 수 없는(Inapplicable) 값을 의미  
한 가지 기억해야 할 것은 NULL은 0(Zero) 또는 공백(Space)과는 다르다  
아래 SQL 문장을 수행해보면 값 이 표시되지 않는 데이터가 NULL이다.

```
SELECT EMPNO, ENAME, SAL, COMM FROM EMP;
```

```
EMPNO  ENAME  SAL      COMM
```

```
-----
7369 SMITH      800
7499 ALLEN     1600      300
```

...

```
7902 FORD      3000
```

```
7934 MILLER    1300
```

NULL이 산술연산에 포함되면 그 결과는 연산에 상관없이 무조건 NULL

```
SELECT EMPNO, ENAME, COMM, COMM + 100 FROM EMP;
```

EMPNO	ENAME	COMM	COMM+100
7369	SMITH	100	200
7499	ALLEN	300	400
7521	WARD	500	600
7566	JONES	(null)	(null)
7654	MARTIN	1400	1500
7698	BLAKE	(null)	(null)
7782	CLARK	(null)	(null)

# NULL 값의 정의 및 처리

1. IS NULL 연산자: NULL값은 값이 없거나, 알 수 없거나, 적용할 수 없다는 의미이므로 NULL값을 조회하고자 할 경우에 사용
2. EMP 테이블에서 comm 이 NULL사원의 empno,ename,job,sal,comm,deptno를 출력

SELECT empno,ename,job,sal,comm,deptno FROM emp WHERE comm IS NULL

Results:

	EMPNO	ENAME	JOB	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	800	(null)	20
2	7566	JONES	MANAGER	2975	(null)	20
3	7698	BLAKE	MANAGER	2850	(null)	30
4	7782	CLARK	MANAGER	2450	(null)	10
5	7788	SCOTT	ANALYST	3000	(null)	20
6	7839	KING	PRESIDENT	5000	(null)	10
7	7876	ADAMS	CLERK	1100	(null)	20
8	7900	JAMES	CLERK	950	(null)	30
9	7902	FORD	ANALYST	3000	(null)	20
10	7934	MILLER	CLERK	1300	(null)	10

# NULL 값의 정의 및 처리

## NVL 함수

1. null값을 특정한 값(실제 값)으로 변환하는데 사용
2. 사용될 수 있는 데이터 타입은 날짜, 문자, 숫자
3. NVL 함수를 사용할 때 전환되는 값의 데이터 타입을 일치

## syntax

1. NVL(expr1,expr2)

1) expr1

null 값을 포함하고 있는 Column이나 표현식

2) expr2

null 변환을 위한 목표 값

## 다양한 데이터형에 대한 NVL변형

❖ 데이터형

변환 예

NUMBER

NVL(comm, 0)

DATE

NVL(hiredate, '01-JAN-99')

CHAR or VARCHAR2

NVL(job, '업무없음')

# NULL 값의 정의 및 처리

1. emp 테이블에서 ename, sal, comm, sal\*12+comm을 출력
2. 단 comm이 null 이면 0로 계산

Sql>SELECT ename,sal,comm,sal\*12+NVL(comm,0) FROM emp

	ENAME	SAL	COMM	SAL*12+NVL(COMM,0)
1	SMITH	800	(null)	9600
2	ALLEN	1600	300	19500
3	WARD	1250	500	15500
4	JONES	2975	(null)	35700
5	MARTIN	1250	1400	16400
6	BLAKE	2850	(null)	34200
7	CLARK	2450	(null)	29400
8	SCOTT	3000	(null)	36000
9	KING	5000	(null)	60000
10	TURNER	1500	0	18000
11	ADAMS	1100	(null)	13200
12	JAMES	950	(null)	11400
13	FORD	3000	(null)	36000
14	MILLER	1300	(null)	15600

# NULL 값의 정의 및 처리

## ■ NVL2

함수의 첫 번째 인자가 NULL이 아니면 두 번째 인자로, NULL이면 세 번째 인자를 리턴한다.

```
SQL> SELECT EMPNO, ENAME, SAL, COMM, SAL+COMM, NVL2(COMM, SAL+COMM, SAL)
2 FROM EMP;
```

EMPNO	ENAME	SAL	COMM	SAL+COMM	NVL2(COMM, SAL+COMM, SAL)
7369	SMITH	800			800
7499	ALLEN	1600	300	1900	1900
7521	WARD	1250	500	1750	1750
7566	JONES	2975			2975
7934	MILLER	1300			1300

## ■ NULLIF

함수의 첫 번째 인자와 두 번째 인자가 같으면 NULL을, 다르면 첫 번째 인자를 리턴한다.

```
SQL> SELECT EMPNO, ENAME, JOB, NULLIF(LENGTH(ENAME), LENGTH(JOB))
2 FROM EMP;
```

EMPNO	ENAME	JOB	NULLIF(LENGTH(ENAME), LENGTH(JOB))
7369	SMITH	CLERK	
7499	ALLEN	SALESMAN	5
7521	WARD	SALESMAN	4
7900	JAMES	CLERK	
7902	FORD	ANALYST	4
7934	MILLER	CLERK	6

# NULL 값의 정의 및 처리

## Coalesce

함수의 이자들 중에 null이 아닌 최초의 인자를 return 한다

```
SQL> SELECT EMPNO, ENAME, SAL, COMM, COALESCE(SAL, COMM, 10)  
2 FROM EMP;
```

EMPNO	ENAME	SAL	COMM	COALESCE(SAL, COMM, 10)
7369	SMITH	800		800
7499	ALLEN	1600	300	1600
7521	WARD	1250	500	1250
...				
7934	MILLER	1300		1300

# 별칭(Alias)

## 열에 별칭(Alias) 부여

1. 질의의 결과를 출력할 때 SQL\*Plus는 열 Heading으로 선택된 열 이름을 사용
2. 이 Heading은 때로 사용자가 이해하기가 어려운 경우가 있기 때문에 열 Heading을 변경하여 질의 결과를 출력하면 보다 사용자가 이해

## 열 별칭(Alias) 정의

1. 열 Heading이름을 변경 합니다.
2. 계산에 유용합니다.
3. 열 이름 바로 뒤에 사용합니다.
4. 열 이름과 별칭 사이에 키워드 AS를 넣기도 합니다.
5. 공백이나 특수 문자 또는 대문자가 있으면 이중 인용부호(" ")가 필요 합니다.

# 별칭(Alias)

EMP 테이블에서 ENAME를 이름으로 SAL을 급여로 출력

Sql> SELECT ename AS 이름, sal 급여 FROM emp

	이름	급여
1	SMITH	800
2	ALLEN	1600
3	WARD	1250
4	JONES	2975
5	MARTIN	1250
6	BLAKE	2850
7	CLARK	2450
8	SCOTT	3000
9	KING	5000
10	TURNER	1500
11	ADAMS	1100
12	JAMES	950
13	FORD	3000
14	MILLER	1300



# 연결 연산자

1. 연결 연산자(||)를 사용하여 문자 표현식을 생성하기 위해 다른 열, 산술 표현식, 상수 값에 열을 연결 할 수 있습니다.
2. 연결자의 왼쪽에 있는 열은 단일 결과 열을 만들기 위해 조합 됩니다.
3. 열이나 문자 STRING을 다른 열에 연결 합니다.
4. 두 개의 "||"로 연결 합니다.
5. 문자 표현식의 결과 열을 생성 합니다.

# 연결 연산자

**EMP 테이블에서 ename과 job을 묶어서 employees로 출력**

```
SELECT ename || ' ' || job AS "employees" FROM emp
```

	employees
1	SMITH CLERK
2	ALLEN SALESMAN
3	WARD SALESMAN
4	JONES MANAGER
5	MARTIN SALESMAN
6	BLAKE MANAGER
7	CLARK MANAGER
8	SCOTT ANALYST
9	KING PRESIDENT
10	TURNER SALESMAN
11	ADAMS CLERK
12	JAMES CLERK
13	FORD ANALYST
14	MILLER CLERK

# 연결 연산자

EMP 테이블에서 ename과 job을 "KING is a PRESIDENT" 형식으로 출력


```
SELECT ename || ' ' || 'is a' || ' ' || job AS "employees Details" FROM emp
```

	employees Details
1	SMITH is a CLERK
2	ALLEN is a SALESMAN
3	WARD is a SALESMAN
4	JONES is a MANAGER
5	MARTIN is a SALESMAN
6	BLAKE is a MANAGER
7	CLARK is a MANAGER
8	SCOTT is a ANALYST
9	KING is a PRESIDENT
10	TURNER is a SALESMAN
11	ADAMS is a CLERK
12	JAMES is a CLERK
13	FORD is a ANALYST
14	MILLER is a CLERK

# 연결 연산자

EMP 테이블에서 **ename**과 **salary**을 “**KING: 1 Year salary = 60000**” 형식으로 출력


```
SELECT ename || ': 1 Year salary = ' || sal * 12 Monthly FROM emp
```

	 MONTHLY
1	SMITH: 1 Year salary = 9600
2	ALLEN: 1 Year salary = 19200
3	WARD: 1 Year salary = 15000
4	JONES: 1 Year salary = 35700
5	MARTIN: 1 Year salary = 15000
6	BLAKE: 1 Year salary = 34200
7	CLARK: 1 Year salary = 29400
8	SCOTT: 1 Year salary = 36000
9	KING: 1 Year salary = 60000
10	TURNER: 1 Year salary = 18000
11	ADAMS: 1 Year salary = 13200
12	JAMES: 1 Year salary = 11400
13	FORD: 1 Year salary = 36000
14	MILLER: 1 Year salary = 15600

# DISTINCT


1. 특별히 명시되지 않았다면, SQL\*Plus는 중복되지는 행을 제거하지 않고 Query 결과를 출력
2. 결과에서 중복되는 행을 제거하기 위해서는 SELECT 키워드 바로 뒤에 DISTINCT를 기술
3. DISTINCT라는 키워드는 항상 SELECT 바로 다음에 기술
4. DISTINCT뒤에 나타나는 칼럼들은 모두 DISTINCT의 영향을 받음
5. DISTINCT뒤에 여러 개의 칼럼을 기술하였을 때 나타나는 행은 칼럼의 조합들이 중복되지 않게 출력
6. DISTINCT를 사용하여 나타나는 결과는 기본적으로 오름차순 정렬
7. EMP 테이블에서 JOB을 모두 출력

Select job from emp;

	 JOB
1	CLERK
2	SALESMAN
3	SALESMAN
4	MANAGER
5	SALESMAN
6	MANAGER
7	MANAGER
8	ANALYST
9	PRESIDENT
10	SALESMAN
11	CLERK
12	CLERK
13	ANALYST
14	CLERK



# DISTINCT

- ❖ EMP 테이블에서 JOB을 중복을 제거하고 출력  
SQL> select distinct job from emp;

	 JOB
1	CLERK
2	SALESMAN
3	PRESIDENT
4	MANAGER
5	ANALYST

- ❖ EMP 테이블에서 deptno별로 job를 한번씩 출력

SELECT DISTINCT deptno, job FROM emp

	 DEPTNO	 JOB
1	20	CLERK
2	30	SALESMAN
3	20	MANAGER
4	30	CLERK
5	10	PRESIDENT
6	30	MANAGER
7	10	CLERK
8	10	MANAGER
9	20	ANALYST

# Sql\*plus명령어

명령	설명
SAV[E] <i>filename</i> [.ext] [REP[LACE]] APP[END]]	버퍼의 내용을 외부 파일로 저장함. REPLACE는 지정된 파일명과 같은 파일이 외부에 존재하면 파일을 교체하며, APPEND는 기존의 파일 뒤에 내용을 추가함. 디폴트 확장자는 sql
GET <i>filename</i> [.ext]	외부 파일의 내용을 버퍼로 읽음
STA[RT] <i>filename</i> [.ext]	외부 파일을 실행
@ <i>filename</i>	외부 파일을 실행(START 명령과 동일)
ED[IT]	버퍼를 수정하기 위해 지정된 편집기를 실행하고 버퍼의 내용을 읽음
ED[IT] [ <i>filename</i> [.ext]]	외부 파일을 수정하기 위해 지정된 편집기를 실행
SPO[OL] [ <i>filename</i> [.ext]] OFF OUT	SQL*Plus 내의 모든 실행 결과를 외부 파일로 저장함. OFF는 저장을 종료하며, OUT은 마찬가지로 저장을 종료하고 프린터로 전송
EXIT	SQL*Plus를 종료

```
SQL> SELECT EMPNO, ENAME, JOB, SAL, COMM FROM EMP;
```

EMPNO	ENAME	JOB	SAL	COMM
7369	SMITH	CLERK	800	
7499	ALLEN	SALESMAN	1600	300
7521	WARD	SALESMAN	1250	500
...				
7900	JAMES	CLERK	950	
7902	FORD	ANALYST	3000	
7934	MILLER	CLERK	1300	

14 개의 행이 선택되었습니다.

```
SQL> SAVE C:\EMP.SQL  
file C:\EMP.SQL(이)가 생성되었습니다  
SQL> START C:\EMP.SQL
```

EMPNO	ENAME	JOB	SAL	COMM
7369	SMITH	CLERK	800	
7499	ALLEN	SALESMAN	1600	300
7521	WARD	SALESMAN	1250	500
...				
7900	JAMES	CLERK	950	
7902	FORD	ANALYST	3000	
7934	MILLER	CLERK	1300	



SQL> SPOOL C:\EMP.TXT

SQL> SELECT EMPNO, ENAME, JOB, SAL FROM EMP;

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7902	FORD	ANALYST	3000
7934	MILLER	CLERK	1300

14 개의 행이 선택되었습니다.

SQL> SPOOL OFF

# 연습문제

1. EMP 테이블의 구조 조회
2. EMP 테이블의 모든 내용을 조회
3. EMP 테이블에서 중복되지 않는 deptno를 출력
4. EMP 테이블의 ename과 job를 연결하여 출력
5. DEPT 테이블의 deptno과 loc를 연결하여 출력
6. EMP 테이블의 job과 sal를 연결하여 출력
7. 사원테이블에서 직원들의 연봉(급여 \* 12) "연 봉"으로 계산하여  
사원명, 급여, 연봉 출력

# 연습문제

사원 테이블을 이용하여 다음과 같은 결과를 얻을 수 있는 SQL 문장을 작성하라.

## 사원정보

---

SMITH의 업무는 CLERK이고 급여는 800만원입니다  
ALLEN의 업무는 SALESMAN이고 급여는 1600만원입니다  
WARD의 업무는 SALESMAN이고 급여는 1250만원입니다  
JONES의 업무는 MANAGER이고 급여는 2975만원입니다  
MARTIN의 업무는 SALESMAN이고 급여는 1250만원입니다  
BLAKE의 업무는 MANAGER이고 급여는 2850만원입니다  
CLARK의 업무는 MANAGER이고 급여는 2450만원입니다  
SCOTT의 업무는 ANALYST이고 급여는 3000만원입니다  
KING의 업무는 PRESIDENT이고 급여는 5000만원입니다  
TURNER의 업무는 SALESMAN이고 급여는 1500만원입니다  
ADAMS의 업무는 CLERK이고 급여는 1100만원입니다  
JAMES의 업무는 CLERK이고 급여는 950만원입니다  
FORD의 업무는 ANALYST이고 급여는 3000만원입니다  
MILLER의 업무는 CLERK이고 급여는 1300만원입니다

14 개의 행이 선택되었습니다.

# 특정 행 검색

1. 일반적인 경우 테이블에 있는 모든 자료를 조회할 필요 없이 사용자가 원하는 자료를 조회하는 경우가 대부분 입니다. 이러한 질의를 만족하게 하는 것이 WHERE절입니다. WHERE절은 수행될 조건 절을 포함하며 FROM절 바로 다음에 기술됩니다.

## 2. Syntax

```
SELECT          [DISTINCT]          {*, column [alias], ... }  
FROM            table_name  
[WHERE          condition]  
[ORDER BY      {column, expression} [ASC | DESC]];
```

- 1) DISTINCT 중복 행 제거 옵션
- 2) \* 테이블의 모든 column 출력
- 3) alias 해당 column에 대한 다른 이름 부여
- 4) table\_name 테이블명 질의 대상 테이블 이름
- 5) WHERE 조건을 만족하는 행들만 검색
- 6) condition column명, 표현식, 문자 상수, 숫자 상수, 비교 연산자로 구성된다.
- 7) ORDER BY 질의 결과 정렬을 위한 옵션(ASC:오름차순(Default),DESC내림차순)

# 특정 행 검색

EMP 테이블에서 hiredate가 1982년 01월 01일 이후 인 사원의 empno,ename,job,sal,hiredate,deptno 을 출력

- ❖ 날짜 - to\_date('2008/04/14 22:02:14', 'yyyy/mm/dd hh24:mi:ss')  
'08/04/14'
- ❖ 오늘 날짜 및 시간 - sysdate

```
select empno,ename,job,sal,hiredate,deptno from emp  
where hiredate >= to_date('1982/01/01', 'yyyy/mm/dd')  hiredate >= '82/01/01'
```

Results:

	EMPNO	ENAME	JOB	SAL	HIREDATE	DEPTNO
1	7788	SCOTT	ANALYST	3000	87/04/19	20
2	7876	ADAMS	CLERK	1100	87/05/23	20
3	7934	MILLER	CLERK	1300	82/01/23	10

# SQL연산자

## SQL연산자

연산자	설 명
BETWEEN a AND b	a와b사이에 있다.(a, b값 포함)
IN (list)	list의 값 중 어느 하나와 일치한다.
LIKE	문자 형태와 일치한다.(%,_사용)
IS NULL	NULL값을 가졌다.
NOT BETWEEN a AND b	a와b사이에 있지않다.(a, b값 포함하지 않음)
NOT IN (list)	list의 값과 일치하지 않는다..
NOT LIKE	문자 형태와 일치하지 않는다.
NOT IS NULL	NULL값을 갖지 않는다.

- BETWEEN연산자: 두 값의 범위에 해당하는 행을 출력하기 위해 사용
- 작은 값을 앞에 기술하고 큰 값은 뒤에 기술

# Between A and B 연산자 사용

- Use the BETWEEN operator to display rows based on a range of values.

```
SQL> SELECT  ename, sal
      2  FROM    emp
      3  WHERE   sal BETWEEN 1000 AND 1500;
```

ENAME	SAL
-----	-----
MARTIN	1250
TURNER	1500
WARD	1250
ADAMS	1100
MILLER	1300

Lower  
limit

Higher  
limit

# Between A and B 연산자 사용

EMP 테이블에서 hiredate가 1982년인 사원의  
empno,ename,job,sal,hiredate,deptno 를 출력

```
SELECT empno,ename,job,sal,hiredate,deptno  
FROM emp  
where hiredate between '82/01/01' and '82/12/31';
```

Results:

	EMPNO	ENAME	JOB	SAL	HIREDATE	DEPTNO
1	7934	MILLER	CLERK	1300	82/01/23	10



# IN 연산자

IN 연산자: 목록에 있는 값에 대해서 출력하기 위해 IN연산자를 사용  
EMP 테이블에서 empno가 7902,7788,7566인 사원의 empno,ename,job,sal,hiredate  
를 출력

```
SELECT empno,ename,job,sal,hiredate FROM emp WHERE empno IN  
(7902,7788,7566)
```

Results:

	EMPNO	ENAME	JOB	SAL	HIREDATE
1	7566	JONES	MANAGER	2975	81/04/02
2	7788	SCOTT	ANALYST	3000	87/04/19
3	7902	FORD	ANALYST	3000	81/12/03

값의 목록을 검색 조건으로 사용

```
SELECT empno, ename, sal, job FROM emp  
WHERE job IN ('MANAGER', 'ANALYST')
```

# 문자열 비교 사용 (LIKE)

1. String 값의 비교에서 wildcard 를 사용할 수 있다.
2. % (percent character)는 문자가 없거나 string을 의미한다.
  - 1) (underscore character)는 한 문자를 의미한다.
  - 2) name에 값이 X\_Y가 포함되어 있는 문자열을 조회하고자 할 경우 Escape를 사용
  - 3) WHERE name LIKE '%XW\_Y%' ESCAPE 'W';

```
SELECT empno, ename, mgr, sal  
FROM emp  
WHERE ename Like 'W%';
```

```
SELECT empno, ename, mgr, sal  
FROM emp  
WHERE ename Like '%N';
```

```
SELECT empno, ename, mgr, sal  
FROM emp  
WHERE ename Like '_A%';
```

# 문자열 비교 사용 (LIKE)

EMP 테이블에서 hiredate가 1982년인 사원의  
empno,ename,job,sal,hiredate,deptno 를 출력

```
SELECT empno,ename,job,sal,hiredate,deptno FROM emp  
where hiredate LIKE '82%'
```

Results:

	EMPNO	ENAME	JOB	SAL	HIREDATE	DEPTNO
1	7934	MILLER	CLERK	1300	82/01/23	10

# 연산자 우선순위

Order Evaluated	Operator
1	All comparison operators
2	NOT
3	AND
4	OR

- Override rules of precedence by using parentheses.

# 연산자 우선순위

```
SQL> SELECT ename, job, sal
  2   FROM    emp
  3   WHERE   job='SALESMAN'
  4   OR      job='PRESIDENT'
  5   AND     sal>1500;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
WARD	SALESMAN	1250

# 연산자 우선순위

```
SQL> SELECT      ename, job, sal
  2  FROM          emp
  3  WHERE         (job=' SALESMAN '
  4  OR           job=' PRESIDENT ' )
  5  AND          sal>1500;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
ALLEN	SALESMAN	1600

# 연산자 우선순위

논리 연산자는 NOT, AND, OR 순의 우선순위가 있으며 우선순위를 임의로 지정하고자 하는 경우에는 괄호를 사용하면 된다. 다음 예제를 보고 어떤 조건이 먼저 적용될 것인지를 생각해 보자.

```
SQL> SELECT EMPNO, ENAME, JOB, SAL
2  FROM EMP
3  WHERE JOB = 'SALESMAN'
4  OR JOB = 'MANAGER'
5  AND SAL >= 2000;
```

EMPNO	ENAME	JOB	SAL
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7844	TURNER	SALESMAN	1500

7 개의 행이 선택되었습니다.

# Order By

1. 질의 결과에 RETURN되는 행의 순서는 정의되지 않습니다.
2. ORDER BY절은 행을 정렬하는데 사용할 수 있습니다.
3. ORDER BY절 사용하는 경우 SELECT문의 맨 뒤에 기술되어야 합니다.
4. 정렬을 위한 표현식이나 Alias를 명시할 수 있습니다.

## 5. Syntax

```
SELECT          [DISTINCT]          {*, column [alias], ...}
FROM            table_name
[WHERE          condition]
[ORDER BY      {column, expression} [ASC | DESC]];
```

ORDER BY	검색된 행이 출력되는 순서를 명시합니다.
ASC	행의 오름차순 정렬(Default)
DESC	행의 내림차순 정렬

6. 디폴트 정렬은 오름차순입니다:
7. 숫자 값은 가장 적은 값이 먼저 출력됩니다.(예 : 1 ~ 999)
8. 날짜 값은 가장 빠른 값이 먼저 출력됩니다.(예 : 01-JAN-92 ~ 01-JAN-95)
9. 문자 값은 알파벳 순서로 출력됩니다.(예 : A ~ Z ~ a ~ z)
10. Null값은 오름차순에서는 제일 나중에 그리고 내림차순에서는 제일 먼저 옵니다.
11. 행이 디스플레이 되는 순서를 바꾸기 위해서, ORDER BY절에서 열 이름 뒤에 DESC키워드를 명시해야 합니다.



# Order By

emp 테이블에서 hiredate의 오름차순으로  
hiredate,empno,ename,job,sal,deptno를 출력

```
SELECT hiredate,empno,ename,job,sal,deptno FROM emp  
ORDER BY hiredate
```

Results:

	HIREDATE	EMPNO	ENAME	JOB	SAL	DEPTNO
1	80/12/17	7369	SMITH	CLERK	800	20
2	81/02/20	7499	ALLEN	SALESMAN	1600	30
3	81/02/22	7521	WARD	SALESMAN	1250	30
4	81/04/02	7566	JONES	MANAGER	2975	20
5	81/05/01	7698	BLAKE	MANAGER	2850	30
6	81/06/09	7782	CLARK	MANAGER	2450	10
7	81/09/08	7844	TURNER	SALESMAN	1500	30
8	81/09/28	7654	MARTIN	SALESMAN	1250	30
9	81/11/17	7839	KING	PRESIDENT	5000	10
10	81/12/03	7900	JAMES	CLERK	950	30
11	81/12/03	7902	FORD	ANALYST	3000	20
12	82/01/23	7934	MILLER	CLERK	1300	10
13	87/04/19	7788	SCOTT	ANALYST	3000	20
14	87/05/23	7876	ADAMS	CLERK	1100	20

# Order By

emp 테이블에서 hiredate의 내림차순으로 hiredate,empno,ename,job,sal,deptno를 출력

```
SELECT hiredate,empno,ename,job,sal,deptno FROM emp  
ORDER BY hiredate desc
```

Results:

	HIREDATE	EMPNO	ENAME	JOB	SAL	DEPTNO
1	87/05/23	7876	ADAMS	CLERK	1100	20
2	87/04/19	7788	SCOTT	ANALYST	3000	20
3	82/01/23	7934	MILLER	CLERK	1300	10
4	81/12/03	7902	FORD	ANALYST	3000	20
5	81/12/03	7900	JAMES	CLERK	950	30
6	81/11/17	7839	KING	PRESIDENT	5000	10
7	81/09/28	7654	MARTIN	SALESMAN	1250	30
8	81/09/08	7844	TURNER	SALESMAN	1500	30
9	81/06/09	7782	CLARK	MANAGER	2450	10
10	81/05/01	7698	BLAKE	MANAGER	2850	30
11	81/04/02	7566	JONES	MANAGER	2975	20
12	81/02/22	7521	WARD	SALESMAN	1250	30
13	81/02/20	7499	ALLEN	SALESMAN	1600	30
14	80/12/17	7369	SMITH	CLERK	800	20

# Order By

## 1. 다양한 정렬 방법

```
SQL> SELECT empno,ename,job,sal,sal*12 annsal FROM emp  
ORDER BY annsal;
```

```
SQL> SELECT empno,ename,job,sal,sal*12 annsal FROM emp ORDER BY sal*12;
```

```
SQL> SELECT empno,ename,job,sal,sal*12 annsal FROM emp ORDER BY 5;
```






2. 하나 이상의 열로 질의 결과를 정렬할 수 있습니다.
3. 주어진 테이블에 있는 개수까지만 가능합니다.
4. ORDER BY절에서 열을 명시하고, 열 이름은 콤마로 구분합니다.
5. 열의 순서를 바꾸고자 한다면 열 이름 뒤에 DESC를 명시합니다.
6. SELECT절에 포함되지 않는 열로 정렬할 수도 있습니다.

# Order By

emp 테이블에서 deptno의 오름차순으로 정렬하고 같은 경우 sal의 내림차순으로 deptno,sal,empno,ename,job 를 출력

```
SELECT deptno,sal,empno,ename,job FROM emp ORDER BY deptno, sal DESC
```

Results :

		DEPTNO		SAL		EMPNO		ENAME		JOB
1		10		5000		7839		KING		PRESIDENT
2		10		2450		7782		CLARK		MANAGER
3		10		1300		7934		MILLER		CLERK
4		20		3000		7788		SCOTT		ANALYST
5		20		3000		7902		FORD		ANALYST
6		20		2975		7566		JONES		MANAGER
7		20		1100		7876		ADAMS		CLERK
8		20		800		7369		SMITH		CLERK
9		30		2850		7698		BLAKE		MANAGER
10		30		1600		7499		ALLEN		SALESMAN
11		30		1500		7844		TURNER		SALESMAN
12		30		1250		7654		MARTIN		SALESMAN
13		30		1250		7521		WARD		SALESMAN
14		30		950		7900		JAMES		CLERK

# Order By

emp 테이블에서 deptno의 오름차순으로 정렬하고 같은 경우 job의 오름차순으로 job이 같은 경우에는 sal의 내림차순으로 deptno,job,sal,empno,ename,hiredate를 출력

```
SELECT deptno,job,sal,empno,ename,hiredate FROM emp  
ORDER BY deptno,job,sal DESC
```

Results:

	DEPTNO	JOB	SAL	EMPNO	ENAME	HIREDATE
1	10	CLERK	1300	7934	MILLER	82/01/23
2	10	MANAGER	2450	7782	CLARK	81/06/09
3	10	PRESIDENT	5000	7839	KING	81/11/17
4	20	ANALYST	3000	7788	SCOTT	87/04/19
5	20	ANALYST	3000	7902	FORD	81/12/03
6	20	CLERK	1100	7876	ADAMS	87/05/23
7	20	CLERK	800	7369	SMITH	80/12/17
8	20	MANAGER	2975	7566	JONES	81/04/02
9	30	CLERK	950	7900	JAMES	81/12/03
10	30	MANAGER	2850	7698	BLAKE	81/05/01
11	30	SALESMAN	1600	7499	ALLEN	81/02/20
12	30	SALESMAN	1500	7844	TURNER	81/09/08
13	30	SALESMAN	1250	7654	MARTIN	81/09/28
14	30	SALESMAN	1250	7521	WARD	81/02/22

# 연습문제

1. EMP 테이블에서 sal이 3000이상인 사원의 empno, ename, job, sal을 출력하는 SELECT 문장을 작성
2. EMP 테이블에서 empno가 7788인 사원의 ename과 deptno를 출력하는 SELECT 문장
3. EMP 테이블에서 hiredate가 1981년 2월 20과 1981년 5월 1일 사이에 입사한 사원의 ename,job,hiredate을 출력하는 SELECT 문장을 작성(단 hiredate 순으로 출력)
4. EMP 테이블에서 deptno가 10,20인 사원의 모든 정보를 출력하는 SELECT 문장을 작성(단 ename순으로 정렬)
5. EMP 테이블에서 sal이 1500이상이고 deptno가 10,30인 사원의 ename과 sal를 출력하는 SELECT 문장을 작성(단 HEADING을 employee과 Monthly Salary로 출력)
6. EMP 테이블에서 hiredate가 1982년인 사원의 모든 정보를 출력하는 SELECT 문을 작성

# 연습문제

1. EMP 테이블에서 COMM에 NULL이 아닌 사원의 모든 정보를 출력하는 SELECT 문을 작성
2. EMP 테이블에서 comm이 sal보다 10% 이상 많은 모든 사원에 대하여 ename,sal, 보너스를 출력하는 SELECT 문을 작성
3. EMP 테이블에서 job이 CLERK이거나 ANALYST이고 sal이 1000,3000,5000이 아닌 모든 사원의 정보를 출력하는 SELECT 문을 작성
4. EMP 테이블에서 ename에 L이 두 자가 있고 deptno가 30이거나 또는 mgr이 7782인 사원의 모든 정보를 출력하는 SELECT 문을 작성하여라.

# 연습문제

1. 사원 테이블에서 입사일이 81년도인 직원의 사번, 사원명, 입사일, 업무, 급여를 검색하시오
2. 사원 테이블에서 입사일이 81년이고 업무가 'SALESMAN'이 아닌 직원의 사번, 사원명, 입사일, 업무, 급여를 검색하시오.
3. 사원 테이블의 사번, 사원명, 입사일, 업무, 급여를 급여가 높은 순으로 정렬하고, 급여가 같으면 입사일이 빠른 사원으로 정렬하시오.
4. 사원 테이블에서 사원명의 세 번째 알파벳이 'N'인 사원의 사번, 사원명을 검색하시오.
5. 사원 테이블에서 연봉( $SAL * 12$ )이 35000 이상인 사번, 사원명, 연봉을 검색하시오.