

Select(SCOTT으로 접속)

샘플 데이터베이스 구조

| | | | | | | | | | |
|---|---------------|-------------|-------------|---------|----------|--------|----------|--|--|
| Objects (Table SCOTT.EMP) | | | | | | | | | |
| SCOTT.EMP | | | | | | | | | |
| EMP: Created: 2010/03/30 오전 11:06:23 Last DDL: 2010/03/30 오전 11:06:23 Primary Key: EMPNO | | | | | | | | | |
| Partitions | Subpartitions | Stats/Size | Referential | Used By | Auditing | | | | |
| Columns | Indexes | Constraints | Triggers | Data | Script | Grants | Synonyms | | |

샘플 데이터베이스 구조

| Table Objects (Table SCOTT.DEPT) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|---------------|-------------|-------------|---------|--------------------|--------|-------------|----|----|-----------|-------|-----------|--------|---|---|---|---|------------|-------|---|--|--|---|--------------------|-----|---|--|--|---|--------------------|
| SCOTT.DEPT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DEPT: Created: 2010/03/30 오전 11:06:23 Last DDL: 2010/03/30 오전 11:06:23 Primary Key: DEPTNO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Partitions | Subpartitions | Stats/Size | Referential | Used By | Auditing | | | | | | | | | | | | | | | | | | | | | | | | | |
| Columns | Indexes | Constraints | Triggers | Data | Script | Grants | | | | | | | | | | | | | | | | | | | | | | | | |
| <table> <tr> <th>Column Name</th> <th>ID</th> <th>PK</th> <th>Index Pos</th> <th>Null?</th> <th>Data Type</th> </tr> <tr> <td>DEPTNO</td> <td>1</td> <td>1</td> <td>1</td> <td>N</td> <td>NUMBER (2)</td> </tr> <tr> <td>DNAME</td> <td>2</td> <td></td> <td></td> <td>Y</td> <td>VARCHAR2 (14 Byte)</td> </tr> <tr> <td>LOC</td> <td>3</td> <td></td> <td></td> <td>Y</td> <td>VARCHAR2 (13 Byte)</td> </tr> </table> | | | | | | | Column Name | ID | PK | Index Pos | Null? | Data Type | DEPTNO | 1 | 1 | 1 | N | NUMBER (2) | DNAME | 2 | | | Y | VARCHAR2 (14 Byte) | LOC | 3 | | | Y | VARCHAR2 (13 Byte) |
| Column Name | ID | PK | Index Pos | Null? | Data Type | | | | | | | | | | | | | | | | | | | | | | | | | |
| DEPTNO | 1 | 1 | 1 | N | NUMBER (2) | | | | | | | | | | | | | | | | | | | | | | | | | |
| DNAME | 2 | | | Y | VARCHAR2 (14 Byte) | | | | | | | | | | | | | | | | | | | | | | | | | |
| LOC | 3 | | | Y | VARCHAR2 (13 Byte) | | | | | | | | | | | | | | | | | | | | | | | | | |

샘플 데이터베이스 구조

| | | | | | | | | |
|---|--|--|--|--|--|--|--|--|
| Database Objects (Table SCOTT.SALGRADE) | | | | | | | <div> <div></div> <div></div> <div></div> </div> | |
| SCOTT.SALGRADE | | | | | | | | |
| <div> <div>+</div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></</div></div> | | | | | | | | |

1.기본적인 Select

❖ SQL SELECT

- ✓ Selection : 질의에 대해 RETURN하고자 하는 테이블의 행을 선택하기 위해 SQL의 Selection기능을 사용할 수 있음
- ✓ Projection : 질의에 대해 RETURN하고자 하는 테이블의 열을 선택하기 위해 SQL의 Projection 기능을 사용할 수 있음
- ✓ Join : 공유 테이블 양쪽의 열에 대해 링크를 생성하여 다른 테이블에 저장되어 있는 데이터를 함께 가져오기 위해 SQL의 join 기능을 사용할 수 있음



1.기본적인 Select

❖ SELECT

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

| | |
|-----------|---|
| SELECT | : 원하는 컬럼을 선택 |
| * | - 테이블의 모든 column 조회 |
| 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) |

1. 기본적인 Select

❖ 모든 열 선택

✓ SELECT 키워드에 “*” 을 사용하여 테이블의 열 데이터 모두를 조회

❖ SCOTT이 소유하고 있는 EMP Table의 모든 데이터를 조회

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 |

1.기본적인 Select

❖ 특정 Column 선택

- ✓ 테이블의 특정 Column을 검색하고자 할 경우 Column이름을 “,”로 구분하여 명시함으로써 특정 Column을 조회
- ✓ 조회 순서는 SELECT문 뒤에 기술한 Column의 순서대로 조회

❖ SCOTT이 소유하고 있는 EMP Table에서 empno, ename, sal, job 를 조회

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 |

1.기본적인 Select

❖ 산술 표현식

- ✓ 데이터가 조회되는 방식을 수정하거나 계산을 수행하고자 할 때 산술 표현식을 사용
- ✓ 산술 표현식은 열 이름, 숫자 상수, 문자 상수, 산술 연산자를 포함할 수 있으며 연산자는 +(Add), -(Subtract), *(Multiply), /(Divide)을 사용
- ✓ SELECT문장에서는 FROM절을 제외한 절에서 사용할 수 있음
- ✓ 산술 표현식이 하나 이상의 연산자를 포함한다면 일반적인 산술 연산자 우선 순위를 적용
- ✓ 날짜는 하루를 숫자 1로 계산해서 덧셈과 뺄셈 가능

❖ emp 테이블의 Sal을 300증가 시키기 위해 덧셈 연산자를 사용하고 ename, sal, sal+300을 조회

```
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 |

1.기본적인 Select

❖ 주의

- ✓ 계산된 결과 열인 SAL+300은 EMP테이블의 새로운 열이 아니고 단지 디스플레이를 위한 것
- ✓ 디폴트로 새로운 열의 이름 sal+300은 생성된 계산식으로부터 유래
- ✓ SQL*Plus는 산술 연산자 앞뒤의 공백을 무시

❖ NULL 값의 처리

- ✓ 행이 특정 열에 대한 데이터 값이 없다면 값은 NULL
- ✓ NULL 값은 이용할 수 없거나 지정되지 않았거나, 알 수 없거나 또는 적용할 수 없는 값
- ✓ NULL 값은 0이나 공백과는 다르며 0은 숫자이며 공백은 문자
- ✓ 열이 NOT NULL로 정의되지 않았거나 열이 생성될 때 PRIMARY KEY로 정의되지 않았다면 열은 NULL 값을 포함할 수 있음
- ✓ NULL 값을 포함한 산술 표현식 결과는 NULL
- ✓ column에 데이터 값이 없으면 그 값 자체가 NULL 또는 NULL 값을 포함
- ✓ NULL 값은 1바이트의 내부 저장 장치를 오버헤드로 사용하고 있으며 어떠한 datatype의 column 들이라도 NULL 값을 포함할 수 있음

1. 기본적인 Select

- ❖ emp 테이블에서 empno, ename, sal, comm, sal+comm/100을 조회
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) |

1. 기본적인 Select

❖ NVL 함수

- ✓ NULL값을 특정한 값(실제 값)으로 변환하는데 사용
- ✓ 사용될 수 있는 데이터 타입은 날짜, 문자, 숫자
- ✓ NVL 함수를 사용할 때 전환되는 값은 컬럼의 데이터 타입을 준수

❖ Syntax

- ✓ NVL(expr1,expr2)
 - expr1 NULL 값을 포함하고 있는 Column이나 표현식
 - expr2 NULL 변환을 위한 목표 값

❖ 다양한 데이터형에 대한 NVL변형

- ✓ 데이터형

NUMBER

DATE

CHAR or VARCHAR2

변환 예

NVL(comm, 0)

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

NVL(job, '업무없음')

1.기본적인 Select

- ❖ emp 테이블에서 ename, sal, comm, sal*12+comm을 조회(단 comm이 NULL 이면 0로 계산)

```
SELECT ename, sal, comm, sal * 12 + NVL(comm,0)
FROM emp;
```

| R2 | ENAME | R2 | SAL | R2 | COMM | R2 | 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 |

1.기본적인 Select

- ❖ 열에 별칭(Alias) 부여: 질의의 결과를 조회할 때 보통 SQL*Plus는 열 Heading으로 선택된 열 이름을 사용하는데 이 Heading은 때로 사용자가 이해하기가 어려운 경우가 있기 때문에 열 Heading을 변경하여 질의 결과를 조회하면 보다 쉽게 사용자가 이해할 수 있음
- ❖ 열 별칭(Alias) 정의
 - ✓ 열 Heading 이름을 변경
 - ✓ 계산식을 조회하는 경우에 유용
 - ✓ SELECT 절에서 열 이름 바로 뒤에 입력
 - ✓ 열 이름과 별칭 사이에 키워드 AS를 넣을 수 있음
 - ✓ 공백이나 특수 문자 또는 대문자가 있으면 이중 인용부호(" ")가 필요



1.기본적인 Select

- ❖ EMP 테이블에서 ENAME를 이름으로 SAL을 급여로 해서 2개 Column의 모든 데이터를 조회

```
SELECT ename AS 이름, sal 급여  
FROM emp
```

| | AZ | 이름 | AZ | 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 |



1.기본적인 Select

❖ 연결 연산자

- ✓ 연결 연산자(||)를 사용하여 문자 표현식을 생성하기 위해 다른 열, 산술 표현식, 상수 값에 열을 연결 할 수 있음
- ✓ 연결자의 왼쪽에 있는 열은 단일 결과 열을 만들기 위해 조합음
- ✓ 열이나 문자 STRING을 다른 열에 연결
- ✓ “||”로 연결



1.기본적인 Select

- ❖ EMP 테이블에서 ename과 job을 묶어서 employees로 조회
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 |



1.기본적인 Select

❖ LITERAL – 문자 STRING

- ✓ LITERAL은 열 이름이나 열 별칭이 아닌 SELECT 목록에 포함되어 있는 문자열, 표현식, 숫자와 같은 상수
- ✓ RETURN되는 각각의 행에 대해 조회
- ✓ LITERAL과 STRING은 질의 결과에 포함될 수 있으며 SELECT목록에서 열과 똑같이 취급
- ✓ 날짜와 문자 LITERAL은 단일 인용 부호(' ')를 사용하여야 하고 숫자 LITERAL은 단일 인용부호를 사용하지 않음



1.기본적인 Select

- ❖ EMP 테이블에서 ename과 job을 “KING is a PRESIDENT” 형식으로 조회
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 |



1.기본적인 Select

- ❖ EMP 테이블에서 ename과 salary을 “KING: 1 Year salary = 60000” 형식으로 조회
SELECT ename || ': 1 Year salary = ' || sal * 12 Monthly
FROM emp;

| R | 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 |




1.기본적인 Select

- ❖ 특별히 명시되지 않았다면, SQL*Plus는 중복되지는 행을 제거하지 않고 Query 결과를 조회
- ❖ 결과에서 중복되는 행을 제거하기 위해서는 SELECT절에서 컬럼 이름 앞에 DISTINCT를 기술
- ❖ DISTINCT라는 키워드는 항상 SELECT 바로 다음에 기술
- ❖ DISTINCT뒤에 나타나는 컬럼들은 모두 DISTINCT의 영향을 받음
- ❖ DISTINCT뒤에 여러 개의 컬럼을 기술하였을 때 나타나는 행은 컬럼의 조합들이 중복되지 않게 조회



1.기본적인 Select

- ❖ EMP 테이블에서 JOB을 모두 조회
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 |



1.기본적인 Select

- ❖ EMP 테이블에서 JOB을 중복을 제거하고 조회
SELECT DISTINCT job
FROM emp;

| EMPID | JOB |
|-------|-----------|
| 1 | CLERK |
| 2 | SALESMAN |
| 3 | PRESIDENT |
| 4 | MANAGER |
| 5 | ANALYST |



1.기본적인 Select

- ❖ EMP 테이블에서 empno별로 job를 한번씩 조회
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 |



2. 연습문제

❖ 아래의 SELECT 문장이 성공적으로 수행 될까요? (참 / 거짓)

```
SELECT ename 이름, job 업무, sal 급여  
FROM emp;
```

❖ 아래의 SELECT 문장이 성공적으로 수행 될까요? (참 / 거짓)

```
SELECT *  
FROM salgrade;
```

❖이 문장에 에러가 있습니다. 올바르게 작성하시오.

```
SELECT empno, ename, sal X 12 년 봉  
FROM emp;
```



2. 연습문제

- ❖ EMP 테이블의 구조 조회
- ❖ EMP 테이블의 모든 데이터를 조회
- ❖ EMP 테이블에서 중복되지 않는 deptno를 조회
- ❖ EMP 테이블의 ename과 job를 연결하여 조회



2.연습문제

❖ 참

❖ 참

```
SELECT empno,ename,sal * 12 as "년 봉"  
FROM emp;
```

```
desc emp;
```

```
SELECT *  
FROM emp;
```

```
SELECT DISTINCT deptno  
FROM emp;
```

```
SELECT ename || job  
FROM emp;
```



3. 특정 행 검색

- ❖ 일반적인 경우 테이블에 있는 모든 자료를 조회할 필요없이 사용자가 원하는 자료를 조회하는 경우가 대부분이며 이러한 질의를 만족하게 하는 것이 WHERE
- ❖ WHERE절은 수행될 조건 절을 포함하며 FROM절 바로 다음에 기술

❖ Syntax

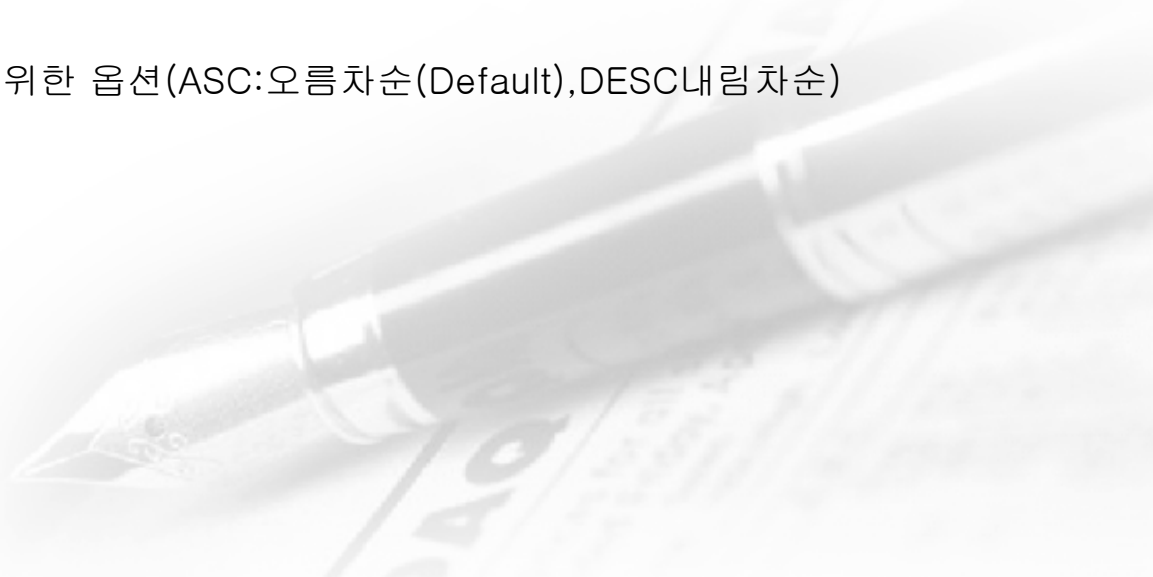
SELECT [DISTINCT] {*, column [alias], ...}

FROM table_name

[WHERE condition]

[ORDER BY {column, expression} [ASC | DESC]];

- ✓ WHERE 행들에 대한 조건을 기술하는 절
 - condition 검색할 조건을 입력: colum명, 표현식, 문자 상수, 숫자 상수, 비교 연산자로 구성
- ✓ ORDER BY 질의 결과 정렬을 위한 옵션(ASC:오름차순(Default),DESC내림차순)



3. 특정 행 검색

❖ WHERE절에 사용되는 연산자

- ✓ WHERE절을 사용하여 행들을 제한
- ✓ WHERE절은 FROM절 다음에 위치
- ✓ 조건은 아래의 것으로 구성
 - column 명, 표현식, 상수
 - 비교 연산자, SQL연산자, 논리연산자
 - 문자(Literal)는 작은 따옴표 안에 표기
 - 날짜도 숫자로 인식하므로 크기 비교 연산 가능

❖ 비교 연산자

| 연산자 | 의미 |
|-------------------|-----------|
| = | 같다 |
| > | 보다 크다 |
| >= | 보다 크거나 같다 |
| < | 보다 작다 |
| <= | 보다 작거나 같다 |
| <>, !=, ^= | 같지 않다 |
| NOT Column_name = | 같지 않다 |
| NOT Column_name > | 보다 크지 않다 |

3. 특정 행 검색

- ❖ EMP 테이블에서 sal이 3000 이상인 사원의 empno, ename, job, sal을 조회

```
SELECT empno, ename, job, sal  
FROM emp  
WHERE sal >= 3000;
```

| | EMPNO | ENAME | JOB | SAL |
|---|-------|-------|-----------|------|
| 1 | 7839 | KING | PRESIDENT | 5000 |
| 2 | 7902 | FORD | ANALYST | 3000 |



3. 특정 행 검색

- ❖ EMP 테이블에서 job이 MANAGER 인 사원의 empno, ename, job, sal, deptno를 조회

```
SELECT empno, ename, job, sal, deptno  
FROM emp  
WHERE job = 'MANAGER'
```

Results:

| | EMPNO | ENAME | JOB | SAL | DEPTNO |
|---|-------|-------|---------|------|--------|
| 1 | 7566 | JONES | MANAGER | 2975 | 20 |
| 2 | 7698 | BLAKE | MANAGER | 2850 | 30 |
| 3 | 7782 | CLARK | MANAGER | 2450 | 10 |



3. 특정 행 검색

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

- ✓ 날짜 - to_date('2008/04/14 22:02:14', 'yyyy/mm/dd hh24:mi:ss')
- ✓ 오늘 날짜 및 시간 - sysdate

```
SELECT empno, ename, job, sal, hiredate, deptno  
FROM emp  
WHERE hiredate >= to_date('1982/01/01', 'yyyy/mm/dd')
```

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



3. 특정 행 검색

❖ SQL연산자

연산자

설 명

AND

두 가지 조건을 모두 만족할 경우에만 검색

OR

두 가지 조건 중 하나라도 만족하는 경우에만 검색

BETWEEN a AND b

a 와 b 사이(a, b값 포함)

IN (list)

list의 값 중 어느 하나와 일치

LIKE

패턴과 일치(%,_사용)

IS NULL

NULL 값 일치

NOT BETWEEN a AND b

a 와 b사이에 있지 않음(a, b값 포함하지 않음)

NOT IN (list)

list의 값과 일치하지 않음

NOT LIKE

패턴과 일치하지 않음

IS NOT NULL

NULL과 일치하지 않음

✓ BETWEEN 연산자

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

3. 특정 행 검색

- ❖ EMP 테이블에서 sal이 1300에서 1500 인 사원의 ename, job, sal, deptno 을 조회

```
SELECT ename,job,sal,deptno  
FROM emp  
WHERE sal BETWEEN 1300 AND 1500;
```

Results:

| | ENAME | JOB | SAL | DEPTNO |
|---|--------|----------|------|--------|
| 1 | TURNER | SALESMAN | 1500 | 30 |
| 2 | MILLER | CLERK | 1300 | 10 |



3. 특정 행 검색

- ❖ IN 연산자: 목록에 있는 값에 대해서 조회하기 위해 IN 연산자를 사용
- ❖ EMP 테이블에서 empno가 7902,7788,7566인 사원의 empno, ename, job, sal, hiredate를 조회

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

| | EMPNO | ENAME | JOB | SAL | HIREDATE |
|---|-------|-------|---------|------|----------|
| 1 | 7566 | JONES | MANAGER | 2975 | 81/04/02 |
| 2 | 7902 | FORD | ANALYST | 3000 | 81/12/03 |

3. 특정 행 검색

❖ LIKE 연산자

- ✓ STRING 값에 대한 와일드 카드 검색을 위해서 LIKE 연산자를 사용
- ✓ 검색 조건은 LITERAL 문자나 숫자를 포함
- ✓ '%'는 문자가 없거나 하나 이상의 문자
- ✓ '_'는 하나의 문자와 대치
- ✓ 패턴 일치 문자를 조합하는 것도 가능
- ✓ '%'나 '_'에 대해서 검색하기 위해서는 ESCAPE 식별자를 이용할 수 있음
- ✓ name에 값이 X_Y가 포함되어 있는 문자열을 조회하고자 할 경우 Escape를 사용
 - WHERE name LIKE '%XW_Y%' ESCAPE 'W';



3. 특정 행 검색

- ❖ EMP 테이블에서 hiredate가 1982년인 사원의 empno, ename, job, sal, hiredate, deptno 를 조회

```
SELECT empno, ename, job, sal, hiredate, deptno  
FROM emp  
WHERE hiredate >= to_date('1982/01/01', 'yyyy/mm/dd') and  
hiredate <= to_date('1982/12/31', 'yyyy/mm/dd');
```

Results:

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

3. 특정 행 검색

- ❖ EMP 테이블에서 hiredate가 1982년인 사원의 empno, ename, job, sal, hiredate, deptno 를 조회

```
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 |



3. 특정 행 검색

- ❖ EMP 테이블에서 hiredate가 12월인 사원의 empno, ename, job, sal, hiredate, deptno 를 조회

```
SELECT empno, ename, job, sal, hiredate, deptno  
FROM emp  
WHERE hiredate LIKE '___12%';
```



3. 특정 행 검색

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

```
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 |

3. 특정 행 검색

- ❖ 논리 연산자: AND, OR, NOT
- ❖ EMP 테이블에서 sal이 2800이상이고 job이 MANAGER 인 사원의 empno, ename, job, sal, hiredate, deptno를 조회

```
SELECT empno, ename, job, sal, hiredate, deptno  
FROM emp  
WHERE sal >= 2800 AND job = 'MANAGER';
```

```
SELECT empno, ename, job, sal, hiredate, deptno  
FROM emp  
WHERE job = 'MANAGER' AND sal >= 2800;
```



3. 특정 행 검색

- ❖ EMP 테이블에서 sal이 1100이상이거나 job이 MANAGER 인 사원의 empno, ename, job, sal, hiredate, deptno를 조회

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

Results:

| | R2 | EMPNO | R2 | ENAME | R2 | JOB | R2 | SAL | R2 | HIREDATE | R2 | 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 |

3. 특정 행 검색

- ❖ NOT 연산자는 BETWEEN, LIKE, IS NULL과 같은 다른 SQL연산자와 함께 사용 가능
- ❖ EMP 테이블에서 job이 MANAGER, CLERK, ANALYST가 아닌 사원의 empno, ename, job, sal, deptno를 조회

```
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 |

3. 특정 행 검색

- ❖ 우선 순위 규칙: 괄호>산술연산자>모든 비교 연산자>NOT>AND>OR
- ❖ emp 테이블에서 job이 SALESMAN 이거나 PRESIDENT이고 sal이 1500이 넘는 사원의 empno, ename, job, sal를 조회

```
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 |



3. 특정 행 검색

- ❖ emp 테이블에서 (job이 SALESMAN 이거나 PRESIDENT)이고 sal이 15000이 넘는 사원의 empno, ename, job, sal를 조회

```
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 |

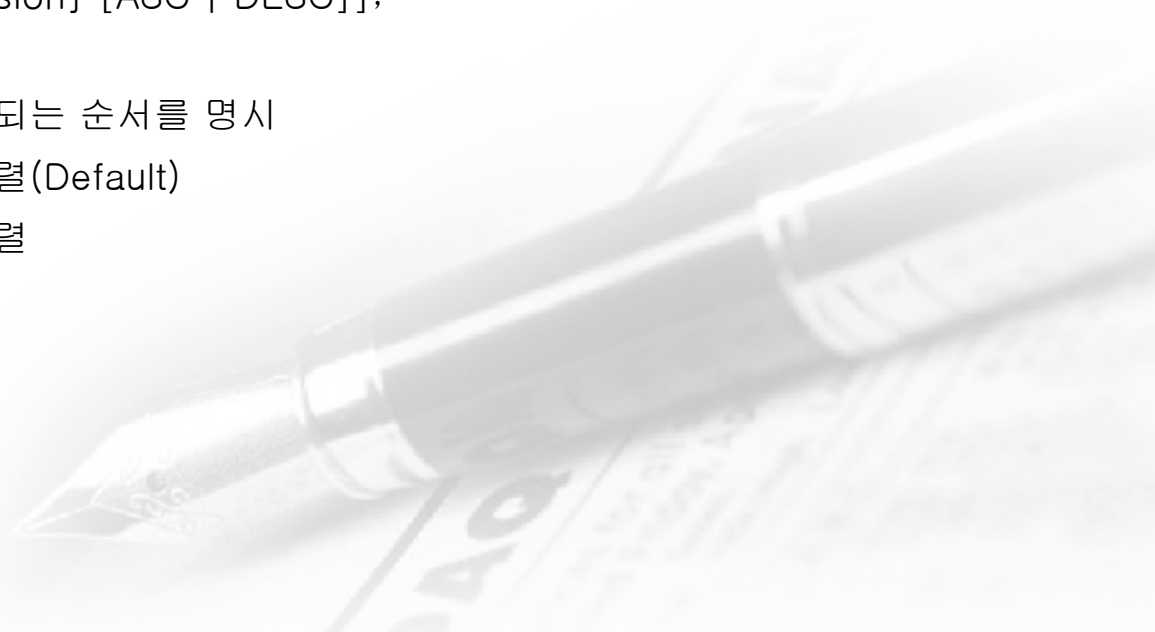
4.Order By

- ❖ SELECT를 수행한 나오는 결과 행의 순서는 알 수 없음
- ❖ ORDER BY절은 행을 정렬하는데 사용
- ❖ ORDER BY절을 사용하는 경우 SELECT문의 맨 뒤에 기술해야 함
- ❖ 정렬을 위한 표현식이나 컬럼의 Alias을 명시할 수 있음

- ❖ Syntax

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

| | |
|----------|---------------------|
| ORDER BY | 검색된 행이 조회되는 순서를 명시 |
| ASC | 행의 오름차순 정렬(Default) |
| DESC | 행의 내림차순 정렬 |



4.Order By

- ❖ 디폴트 정렬은 오름차순
- ❖ 숫자 값은 가장 적은 값이 먼저 조회(예 : 1 ~ 999)
- ❖ 날짜 값은 가장 빠른 값이 먼저 조회(예 : 01-JAN-92 ~ 01-JAN-95)
- ❖ 문자 값은 알파벳 순서로 조회(예 : A ~ Z ~ a ~ z)
- ❖ NULL값은 오름차순에서는 제일 나중에 그리고 내림차순에서는 제일 먼저
- ❖ 행이 디스플레이 되는 순서를 큰 것을 먼저 조회하기 위해서는 ORDER BY절에서 열 이름 뒤에 DESC키워드를 명시



4.Order By

- ❖ emp 테이블에서 hiredate의 오름차순으로 hiredate, empno, ename, job, sal, deptno를 조회

```
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 |

4.Order By

- ❖ emp 테이블에서 hiredate의 내림차순으로 hiredate,empno,ename,job,sal,deptno를 조회

```
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 |

4.Order By

❖ 다양한 정렬 방법

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

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

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

❖ 하나 이상의 열로 질의 결과를 정렬

❖ ORDER BY절에서 열을 명시하고, 열 이름은 콤마로 구분

❖ SELECT절에 포함되지 않는 열이나 계산식으로 정렬 가능



4.Order By

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

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

Results:

| | R 2 | DEPTNO | R 2 | SAL | R 2 | EMPNO | R 2 | ENAME | R 2 | 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 |

4.Order By

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

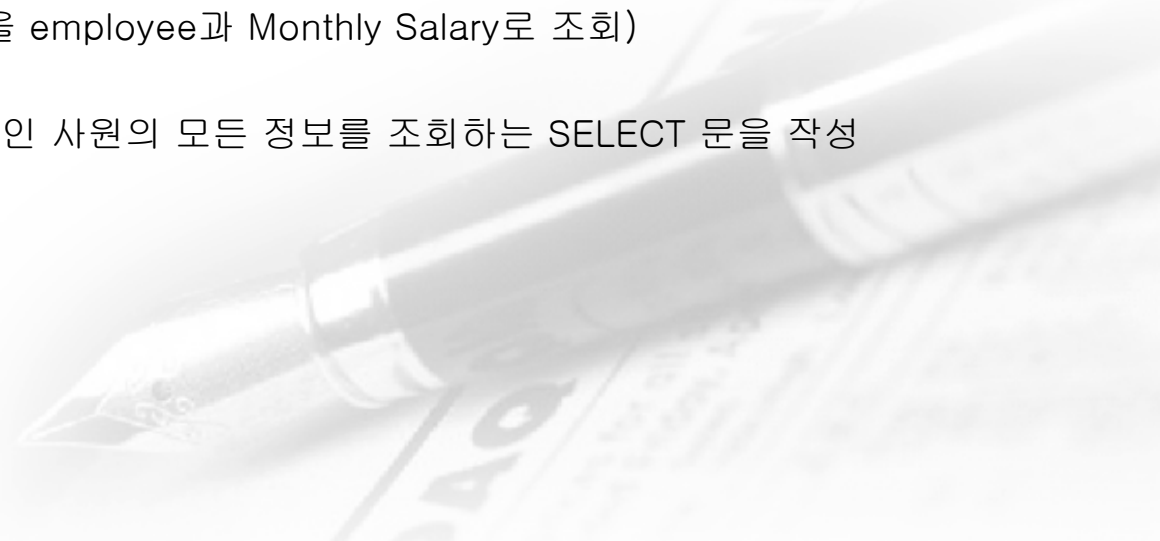
```
SELECT deptno, job, sal, empno, ename, hiredate
FROM emp
ORDER BY deptno, job, sal DESC;
```

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 |

5. 연습문제

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



5. 연습문제

- ❖ EMP 테이블에서 COMM이 NULL이 아닌 사원의 모든 정보를 조회하는 SELECT 문을 작성
- ❖ EMP 테이블에서 comm이 sal보다 10% 이상 많은 사원에 대하여 ename,sal, comm를 조회하는 SELECT 문을 작성
- ❖ EMP 테이블에서 job이 CLERK이거나 ANALYST이고 sal이 1000, 3000, 5000이 아닌 사원의 모든 정보를 조회하는 SELECT 문을 작성
- ❖ EMP 테이블에서 (ename에 L이 두 자 이상이 포함되어 있고 deptno가 30)이거나 mgr이 7782인 사원의 모든 정보를 조회하는 SELECT 문을 작성

