Oracle Database 11g : SQL Fundamentals I

Practice 1

1. DEPARTMENTS 테이블의 구조를 표시하고 테이블의 모든 데이터를 선택하시오.

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

2. EMPLOYEES 테이블의 구조를 표시하시오. 사원 번호가 가장 앞에 오고 이어서 각 사원의 이름, 업무 코드, 입사일이 오도록 질의를 작성하시오. HIRE_DATE 열에 STARTDATE라는 별칭을 지정하시오.

```
SQL> DESCRIBE employees
SQL> SELECT employee_id, last_name, job_id, hire_date StartDate
     FROM   employees;
```

3. EMPLOYEES 테이블의 업무 코드를 중복되지 않게 표시하는 질의를 작성하시오.

```
SQL> SELECT DISTINCT job_id
     FROM   employees;
```

4. 2번의 명령문을 복사하시오. 머리글을 각각 Emp #, Employee, Job 및 Hire Date로 명명한 다음 질의를 다시 실행하시오.

```
SQL> SELECT employee_id "Emp #", last_name "Employee",
            job_id "Job", hire_date "Hire Date"
     FROM   employees
```


5. 업무 ID와 이름을 연결한 다음 쉼표 및 공백으로 구분하여 표시하고 열 이름을 Employee and Title로 지정하시오.

```
SQL> SELECT last_name||', '||job_id "Employee and Title"
     FROM   employees;
```

Practice 2

1. 급여가 12,000를 넘는 사원의 이름과 급여를 표시하는 질의를 실행하시오.

```
SQL> SELECT  last_name, salary
     FROM    employees
     WHERE   salary > 12000;
```

2. 사원 번호가 176인 사원의 이름과 부서 번호를 표시하는 질의를 실행하시오.

```
SQL> SELECT  last_name, department_id
     FROM    employees
     WHERE   employee_id = 176;
```

3. 급여가 5,000에서 12,000 사이에 포함되지 않는 모든 사원의 이름과 급여를 표시하도록 질의를 실행하시오.

```
SQL> SELECT  last_name, salary
     FROM    employees
     WHERE   salary NOT BETWEEN 5000 AND 12000;
```

4. last name이 Matos와 Taylor인 사원의 last_name, 업무 ID, 그리고 입사일을 표시하시오. 결과는 입사일을 기준으로 오름차순 정렬하시오.

```
SQL> SELECT   last_name, job_id, hire_date
     FROM     employees
     WHERE    last_name IN ('Matos','Taylor')
     ORDER BY hire_date;
```

5. 부서 20 및 50에 속하는 모든 사원의 이름과 부서 번호를 이름을 기준으로 영문자순으로 표시하시오.

```
SQL> SELECT   last_name, department_id
     FROM     employees
     WHERE    department_id IN (20, 50)
     ORDER BY last_name;
```

6. 급여가 5,000와 12,000 사이이고 부서 번호가 20 또는 50인 사원의 이름과 급여를 나열하도록 질의를 작성하시오. 열 레이블은 Employee와 Monthly Salary로 각각 지정하시오.

```
SQL> SELECT   last_name "Employee", salary "Monthly Salary"
     FROM     employees
     WHERE    salary BETWEEN 5000 AND 12000
     AND      department_id IN (20, 50);
```

7. 2004년에 입사한 모든 사원의 이름과 입사일을 표시하시오.

```
SQL> SELECT   last_name, hire_date
     FROM     employees
     WHERE    hire_date LIKE '04%';
```

8. 관리자가 없는 모든 사원의 이름과 업무를 표시하시오.

```
SQL> SELECT   last_name, job_id
     FROM     employees
     WHERE    manager_id IS NULL;
```

9. 커미션을 받는 모든 사원의 이름, 급여 및 커미션을 급여 및 커미션을 기준으로 내림차순으로 정렬하여 표시하시오.

```
SQL> SELECT   last_name, salary, commission_pct
     FROM     employees
     WHERE    commission_pct IS NOT NULL
     ORDER BY salary DESC, commission_pct DESC;
```

10. 이름의 세번째 문자가 *a*인 모든 사원의 이름을 표시하시오.

```
SQL> SELECT   last_name
     FROM     employees
     WHERE    last_name LIKE '__a%';
```

11. 이름에 *a*와 *e*가 있는 모든 사원의 이름을 표시하시오.

```
SQL> SELECT last_name FROM employees
      WHERE    last_name LIKE '%a%'
      AND    last_name LIKE '%e%';
```

12. 업무가 영업 사원 또는 사무원이면서 급여가 2,500, 3,500 또는 7,000이 아닌 모든 사원의 이름, 업무 및 급여를 표시하시오.

```
SQL> SELECT   last_name, job_id, salary
      FROM     employees
      WHERE    job_id IN ('SA_REP', 'ST_CLERK')
      AND    salary NOT IN (2500, 3500, 7000);
```

13. 커미션 비율이 20%인 모든 사원의 이름, 급여 및 커미션을 표시하도록 명령문을 작성하여 실행하시오.

```
SQL> SELECT   last_name, salary, commission_pct
     FROM     employees
     WHERE    commission_pct = .20;
```


Practice 3

1. 현재 날짜를 표시하는 질의를 작성하고 열 레이블을 Date로 지정하시오.

```
SQL> SELECT   sysdate "Date" FROM dual;
```

2. 각 사원에 대해 사원 번호, 이름, 급여 및 15% 인상된 급여를 정수로 표시하시오. 인상된 급여 열의 레이블을 New Salary로 지정하시오.

```
SQL> SELECT   employee_id, last_name, salary, ROUND(salary * 1.15, 0) "New Salary"
     FROM     employees;
```

3. 2번 질의를 수정하여 새 급여에서 이전 급여를 빼는 새 열을 추가하고 레이블을 Increase로 지정하고 수정한 질의를 실행하시오.

```
SQL> SELECT   employee_id, last_name, salary, ROUND(salary * 1.15, 0) "New Salary",
          ROUND(salary * 1.15, 0) - salary  AS "Increase"
     FROM     employees;
```

5. 이름이 *J*, *A* 또는 *M*으로 시작하는 모든 사원의 이름(대문자 표시) 및 이름 길이를 표시하는 질의를 작성하고 각 열에 적합한 레이블을 지정하시오. 결과를 사원의 이름에 따라 정렬하시오.

```
SQL> SELECT UPPER(last_name) "Name",LENGTH(last_name) "Length"
     FROM    employees
     WHERE   last_name LIKE 'J%'
     OR      last_name LIKE 'M%'
     OR      last_name LIKE 'A%'
     ORDER BY last_name;
```

6. 각 사원의 이름을 표시하고 근무 달 수(입사일로부터 현재까지의 달 수)를 계산하여 열 레이블을 MONTHS_WORKED로 지정하시오. 결과는 정수로 반올림하여 표시하고 근무 달 수를 기준으로 정렬하시오.

```
SQL> SELECT  last_name, ROUND(MONTHS_BETWEEN (SYSDATE, hire_date)) MONTHS_WORKED
      FROM    employees
    ORDER BY MONTHS_BETWEEN(SYSDATE, hire_date);
```

7. 모든 사원의 성 및 급여를 표시하기 위한 query를 작성합니다. 급여가 15자 길이로 표시되고 왼쪽에 $ 기호가 채워지도록 형식을 지정하시오. 열 레이블을 SALARY 로 지정합니다.

```
SQL> SELECT last_name, LPAD(salary, 15, '$') SALARY
     FROM employees;
```

8. 부서 90의 모든 사원에 대해 성(last_name) 및 재직 기간(주 단위)을 표시하도록 query 를 작성하시오. 주를 나타내는 숫자 열의 레이블로 TENURE를 지정하고 주를 나타내는 숫자 값을 소수점 왼쪽에서 truncate 하시오. 그리고 직원 재직 기간의 내림차순으로 레코드를 표시합니다.

```
SQL> SELECT last_name, trunc((SYSDATE-hire_date)/7) AS TENURE
     FROM employees
     WHERE department_id = 90
     ORDER BY TENURE DESC;
```

Practice 4

1. 각 사원에 대해 다음 항목을 생성하는 질의를 작성하고 열 레이블을 Dream Salaries로 지정하시오.

<employee last name> earns <salary> monthly but wants <3 times salary>.

<예시> Matos earns $2,600.00 monthly but wants $7,800.00.

```
SQL> SELECT  last_name || ' earns ' || TO_CHAR(salary, '$99,999.00') || ' monthly but wants '
             || TO_CHAR(salary * 3, '$99,999.00') || '.' "Dream Salaries"
     FROM    employees;
```

2. 모든 사원의 이름과 급여를 표시하는 질의를 작성하시오. 급여는 15자 길이로 왼쪽에 $ 기호가 채워진 형식으로 표기하고 열 레이블을 SALARY로 지정하시오.

```
SQL> SELECT  last_name, LPAD(salary, 15, '$') SALARY
     FROM    employees;
```

3. 사원의 이름, 입사일 및 급여 검토일을 표시하시오. 급여 검토일은 여섯 달이 경과한 후 첫번째 월요일입니다. 열 레이블을 REVIEW로 지정하고 날짜는 "2010.03.31 월요일"과 같은 형식으로 표시되도록 지정하시오.

```
SQL> SELECT   last_name, hire_date, TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6),'월요일'),
            'YYYY.MM.DD DAY') REVIEW
     FROM     employees;
```

4. 이름, 입사일 및 업무 시작 요일을 표시하고 열 레이블을 DAY로 지정하시오. 월요일을 시작으로 해서 요일을 기준으로 결과를 정렬하시오.

```
SQL> SELECT   last_name, hire_date, TO_CHAR(hire_date, 'DAY') DAY
     FROM     employees
     ORDER BY TO_CHAR(hire_date - 1, 'd');
```

5. 사원의 이름과 커미션을 표시하는 질의를 작성하시오. 커미션을 받지 않는 사원일 경우 "No Commission"을 표시하시오. 열 레이블은 COMM으로 지정하시오.

```
SQL> SELECT   last_name, NVL(TO_CHAR(commission_pct), 'No Commission') COMM
     FROM     employees;
```

6. 하나의 레이블로 사원의 이름을 표시과 급여 총액을 별표(*)로 나타내는 질의를 작성하시오. 각 별표는 1,000달러를 나타냅니다. 급여를 기준으로 데이터를 내림차순으로 정렬하고 열 레이블을 EMPLOYEES_AND_THEIR_SALARIES로 지정하시오.

```
SQL> SELECT   rpad(last_name, 8)||' '|| rpad(' ', salary/1000+1, '*')
               EMPLOYEES_AND_THEIR_SALARIES
     FROM     employees
     ORDER BY salary DESC;
```


7. DECODE 함수와 CASE 구문을 사용하여 다음 데이터에 따라 JOB_ID 열의 값을 기준으로 모든 사원의 등급을 표시하는 질의를 작성하시오.

업무	등급
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
None of the above	0

```
SQL> SELECT job_id, decode (job_id, 'ST_CLERK',  'E',
                                   'SA_REP',    'D',
                                   'IT_PROG',   'C',
                                   'ST_MAN',    'B',
                                    'AD_PRES',   'A',
                                               '0') GRADE
     FROM employees;
```

```
SQL> SELECT job_id, CASE job_id WHEN 'ST_CLERK' THEN 'E'
                                WHEN 'SA_REP'   THEN 'D'
                                WHEN 'IT_PROG'  THEN 'C'
                                WHEN 'ST_MAN'   THEN 'B'
                                WHEN 'AD_PRES'  THEN 'A'
                                ELSE '0'  END  GRADE
     FROM employees;
```

Practice 5

다음 세 문장의 유효성을 판별하여 True 또는 False로 답하시오.

1. 그룹 함수는 여러 행에 적용되어 그룹 당 하나의 결과를 출력한다. (True)

2. 그룹 함수는 계산에 널을 포함한다. (False)

3. WHERE 절은 그룹 계산에 행(row)을 포함시키기 전에 행을 제한한다. (True)

4. 모든 사원의 급여 최고액, 최저액, 총액 및 평균액을 표시하시오. 열 레이블을 각각 Maximum, Minimum, Sum 및 Average로 지정하고 결과를 정수로 반올림하도록 작성하시오.

```
SQL> SELECT   ROUND(MAX(salary),0) "Maximum", ROUND(MIN(salary),0) "Minimum",
              ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average"
     FROM     employees;
```

5. 위의 질의를 수정하여 각 업무 유형(job_id)별로 급여 최고액, 최저액, 총액 및 평균액을 표시하시오.

```
SQL> SELECT   job_id, ROUND(MAX(salary),0) "Maximum",  ROUND(MIN(salary),0) "Minimum",
              ROUND(SUM(salary),0) "Sum", ROUND(AVG(salary),0) "Average"
     FROM     employees
     GROUP BY job_id;
```

6. 업무별 사원 수를 표시하는 질의를 작성하시오.

```
SQL> SELECT   job_id, COUNT(*)
     FROM     employees
     GROUP BY job_id;
```

7. 관리자 수를 확인하시오. 열 레이블은 Number of Managers로 지정하시오. (힌트: MANAGER_ID 열을 사용)

```
SQL> SELECT   COUNT(DISTINCT manager_id) "Number of Managers"
     FROM     employees;
```

8. 최고 급여와 최저 급여의 차액을 표시하는 질의를 작성하고 열 레이블을 DIFFERENCE로 지정하시오.

```
SQL> SELECT   MAX(salary) - MIN(salary) DIFFERENCE
     FROM     employees;
```

9. 관리자 번호 및 해당 관리자에 속한 사원의 최저 급여를 표시하시오. 관리자를 알 수 없는 사원 및 최저 급여가 6,000 미만인 그룹은 제외시키고 결과를 급여에 대한 내림차순으로 정렬하시오.

```
SQL> SELECT   manager_id, MIN(salary) FROM    employees
     WHERE    manager_id IS NOT NULL
     GROUP BY manager_id
     HAVING   MIN(salary) > 6000
     ORDER BY MIN(salary) DESC;
```

10. 업무를 표시한 다음 해당 업무에 대해 부서 번호별 급여 및 부서 20, 50, 80 및 90의 급여 총액을 각각 표시하는 행렬 질의를 작성하고 각 열에 적합한 머리글을 지정하시오.

```
SQL> SELECT   job_id "Job",  SUM(DECODE(department_id , 20, salary)) "Dept 20",
             SUM(DECODE(department_id , 50, salary)) "Dept 50",
             SUM(DECODE(department_id , 80, salary)) "Dept 80",
             SUM(DECODE(department_id , 90, salary)) "Dept 90",
             SUM(salary) "Total"
     FROM    employees
     GROUP BY job_id;
```

Practice 6

1. LOCATIONS 및 COUNTRIES 테이블을 사용하여 HR 부서를 위해 모든 부서의 주소를 생성하는 query를 작성하시오. 출력에 위치 ID, 주소, 구/군, 시/도 및 국가를 표시하며, NATURAL JOIN을 사용하여 결과를 생성합니다.

```
SQL> SELECT location_id, street_address, city, state_province, country_name
     FROM locations NATURAL JOIN countries;
```

2. 모든 사원의 성, 소속 부서번호 및 부서 이름을 표시하는 query를 작성하시오.

```
SQL> SELECT last_name, department_id, department_name
     FROM employees JOIN departments
     USING (department_id);
```

3. Toronto에 근무하는 사원에 대한 보고서를 필요로 합니다. toronto에서 근무하는 모든 사원의 성, 직무, 부서 번호 및 부서 이름을 표시하시오. (힌트 : 3-way join 사용)

```
SQL> SELECT e.last_name, e.job_id, e.department_id, d.department_name
     FROM employees e JOIN departments d
     ON (e.department_id = d.department_id)
     JOIN locations l
     ON (d.location_id = l.location_id)
     WHERE LOWER(l.city) = 'toronto';
```

4. 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성하는데, 열 레이블을 각각 Employee, Emp#, Manager 및 Mgr#으로 지정하시오.

```
SQL> SELECT w.last_name "Employee", w.employee_id "EMP#", m.last_name "Manager",
            m.employee_id "Mgr#"
     FROM employees w join employees m
    ON (w.manager_id = m.employee_id);
```

5. King과 같이 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 4번 문장을 수정합니다. 사원 번호순으로 결과를 정렬하시오.

```
SQL> SELECT w.last_name "Employee", w.employee_id "EMP#", m.last_name "Manager",
       m.employee_id "Mgr#"
       FROM employees w LEFT OUTER JOIN employees m
       ON (w.manager_id = m.employee_id)
       ORDER BY 2;
```

6. 사원의 성과 부서 번호 및 주어진 사원과 동일한 부서에 근무하는 모든 사원을 표시하는 보고서를 작성하시오. 각 열에 적절한 레이블을 자유롭게 지정해 봅니다.

```
SQL> SELECT e.department_id department, e.last_name employee, c.last_name colleague
     FROM employees e JOIN employees c
     ON (e.department_id = c.department_id)
     WHERE e.employee_id <> c.employee_id
     ORDER BY e.department_id, e.last_name, c.last_name;
```


7. HR 부서에서 직무 등급 및 급여에 대한 보고서를 필요로 합니다. 먼저 JOB_GRADES 테이블의 구조를 표시한 다음 모든 사원의 이름, 직무, 부서 이름, 급여 및 등급을 표시하는 query를 작성하시오.

```
SQL> DESC JOB_GRADES
SQL> SELECT e.last_name, e.job_id, d.department_name, e.salary, j.grade_level
     FROM employees e JOIN departments d
     ON (e.department_id = d.department_id)
    JOIN job_grades j
     ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

Practice 7

1. Zlotkey와 동일한 부서에 속한 모든 사원의 이름과 입사일을 표시하는 질의를 작성하시오. Zlotkey는 결과에서 제외하시오.

```
SQL> SELECT last_name, hire_date
    FROM   employees
    WHERE  department_id = (SELECT department_id FROM   employees
                            WHERE  last_name = 'Zlotkey')
    AND    last_name <> 'Zlotkey';
```

2. 급여가 평균 급여보다 많은 모든 사원의 사원 번호와 이름을 표시하는 질의를 작성하고 결과를 급여에 대해 오름차순으로 정렬하시오.

```
SQL> SELECT employee_id, last_name FROM   employees
     WHERE  salary > (SELECT AVG(salary) FROM   employees)
     ORDER BY salary;
```

3. 이름에 *u*가 포함된 사원과 같은 부서에서 일하는 모든 사원의 사원 번호와 이름을 표시하는 질의를 작성하고 질의를 실행하시오.

```
SQL> SELECT employee_id, last_name
     FROM   employees
     WHERE department_id IN (SELECT department_id FROM   employees
                             WHERE last_name like '%u%');
```

4. 부서 위치 ID가 1700인 모든 사원의 이름, 부서 번호 및 업무 ID를 표시하시오.

```
SQL> SELECT last_name, department_id, job_id FROM   employees
     WHERE department_id IN (SELECT department_id FROM   departments
                             WHERE location_id = 1700);
```

5. King에게 보고하는(manager가 King) 모든 사원의 이름과 급여를 표시하시오.

```
SQL> SELECT last_name, salary
     FROM   employees
     WHERE  manager_id = (SELECT employee_id FROM   employees
                          WHERE  last_name = 'King');
```


6. Executive 부서의 모든 사원에 대한 부서 번호, 이름 및 업무 ID를 표시하시오.

```
SQL> SELECT department_id, last_name, job_id
     FROM   employees
     WHERE  department_id IN (SELECT department_id FROM   departments
                              WHERE  department_name = 'Executive');
```

7. 평균 급여보다 많은 급여를 받고 이름에 *u*가 포함된 사원과 같은 부서에서 근무하는 모든 사원의 사원 번호, 이름 및 급여를 표시하시오.

```
SQL> SELECT employee_id, last_name, salary
     FROM   employees
     WHERE  department_id IN (SELECT department_id FROM   employees
                              WHERE  last_name like '%u%')
     AND    salary > (SELECT AVG(salary) FROM   employees);
```

Practice 8

1. SET 연산자를 사용하여 업무 ID ST_CLERK을 포함하지 않는 부서의 ID를 나열하시오.

```
SQL> SELECT department_id FROM departments
     MINUS
     SELECT department_id FROM employees
     WHERE job_id = 'ST_CLERK';
```

2. SET 연산자를 사용하여 해당 지역에 부서가 없는 지역 ID와 지역 이름을 표시하시오.

```
SQL> SELECT country_id, country_name FROM  countries
      MINUS
      SELECT l.country_id, c.country_name
       FROM  locations l JOIN countries c
      ON (l.country_id = c.country_id);
```

3. 입사 이후 현재 업무와 같은 업무를 담당한 적이 있는 사원(업무가 변경되었다가 현재의 업무로 복귀된 사원)의 사원 ID와 업무 ID를 나열하시오.

```
SQL> SELECT   employee_id,job_id FROM employees
     INTERSECT
     SELECT  employee_id,job_id FROM   job_history;
```

4. 다음을 모두 나열하는 혼합 질의를 작성하시오.
   - 소속된 부서에 상관없이 EMPLOYEES 테이블에 있는 모든 사원의 이름과 부서 ID
   - 소속된 사원에 상관없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID와 부서 이름

```
SQL> SELECT last_name, department_id, TO_CHAR(null) FROM  employees
     UNION
     SELECT TO_CHAR(null), department_id, department_name FROM  departments;
```

Practice 9

1. 다음과 같이 실습에 사용할 MY_EMPLOYEE 테이블을 생성하시오.

```
SQL> CREATE TABLE my_employee
     (id  NUMBER(4) CONSTRAINT my_employee_id_nn NOT NULL,
      last_name VARCHAR2(25),
      first_name VARCHAR2(25),
      userid  VARCHAR2(8),
      salary  NUMBER(9,2));
```

2. MY_EMPLOYEE 테이블의 구조를 표시하여 열 이름을 식별하시오.

```
SQL> DESCRIBE my_employee
```

3. 다음 예제 데이터의 첫번째 데이터 행(row)을 MY_EMPLOYEE 테이블에 추가하시오. 이때 INSERT 절에 열을 나열하지 마시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	Rpatel	895
2	Dancs	Betty	Bdancs	860
3	Biri	Ben	Bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropeburn	Audery	Aropebur	1550

```
SQL> INSERT INTO my_employee
     VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

4. 위의 목록에 있는 예제 데이터의 두 번째 행을 MY_EMPLOYEE 테이블에 추가하시오. 이번에는 INSERT 절에 열을 명시적으로 나열하시오.

```
SQL> INSERT INTO my_employee (id, last_name, first_name, userid, salary)
     VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5. 테이블에 추가한 항목을 확인하시오.

```
SQL> SELECT   *  FROM    my_employee;
```

6. loademp.sql이라는 텍스트 파일에 MY_EMPLOYEE 테이블로 행을 로드하는 insert 문을 작성하시오. 이름의 첫 글자와 성의 처음 일곱 글자를 연결하여 사용자 ID를 만드시오. 메모장을 열어 다음 명령문을 입력한 후 loademp.sql로 저장하시오.(위치 : c:\db_test\sql_labs)

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       lower(substr('&p_first_name', 1, 1) ||
       substr('&p_last_name', 1, 7)), &p_salary);
```

7. 작성한 스크립트의 insert 문을 실행하여 예제 데이터의 3,4번의 두 행(row)을 테이블에 추가하시오.

```
SQL> @ c:\db_test\sql_labs\loademp.sql → 프롬프트에 값을 입력
```

8. 테이블에 추가한 항목을 확인하시오.

```
SQL> SELECT   *   FROM my_employee;
```

9. 추가한 데이터를 영구히 저장하시오.

```
SQL> COMMIT;
```

10. 사원 3의 성을 Drexler로 변경하시오.

```
SQL> UPDATE  my_employee
     SET    last_name = 'Drexler'
     WHERE   id = 3;
```

11. 급여가 900 미만인 모든 사원의 급여를 1000으로 변경하고 테이블의 변경 내용을 확인하시오.

```
SQL> UPDATE  my_employee
     SET     salary = 1000
     WHERE   salary < 900;
SQL> SELECT  last_name, salary FROM    my_employee;
```

12. MY_EMPLOYEE 테이블에서 Betty Dancs를 삭제하고 테이블의 변경 내용을 확인하시오.

```
SQL>  DELETE FROM  my_employee
     WHERE last_name = 'Dancs';
SQL> SELECT  * FROM    my_employee;
```

13. 보류 중인 변경 내용을 모두 커밋하시오.

```
SQL> COMMIT;
```

14. 6 단계에서 작성한 스크립트의 명령문을 실행하여 예제 데이터의 마지막 행(row)을 테이블에 추가하시오.

```
SQL>  @c:\db_test\sql_labs loademp.sql
```

15. 테이블에 추가한 항목을 확인하시오.

```
SQL> SELECT * FROM my_employee;
```

16. 트랜잭션 수행 중에 Savepoint를 표시하시오.

```
SQL> SAVEPOINT step_16;
```

17. 테이블의 내용을 모두 삭제하고 테이블 내용이 비어 있는지 확인하시오.

```
SQL> DELETE  FROM  my_employee;
SQL> SELECT * FROM   my_employee;
```

18. 이전의 INSERT 작업은 버리지 말고 최근의 DELETE 작업만 버리시오.

```
SQL> ROLLBACK TO step_16;
```

19. 새 행이 그대로 있는지 확인하시오.

```
SQL> SELECT * FROM   my_employee;
```

20. 추가한 데이터를 영구히 저장하시오.

```
SQL> COMMIT;
```

Practice 10

1. 기존의 DEPT 테이블을 삭제하고 다음 테이블 인스턴트 차트를 기반으로 DEPT 테이블을 생성하시오. 테이블을 생성한 후, 테이블이 생성되었는지 확인하시오.

열이름	ID	NAME

키유형	Primary Key	
널/고유		
FK테이블		
FK 열		
데이터유형	NUMBER	VARCHAR2
길이	7	25

```
SQL> DROP TABLE dept PURGE;
SQL> CREATE TABLE dept
       (id NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
        name VARCHAR2(25));
SQL> DESCRIBE dept
```

2. DEPARTMENT 테이블의 선택된 열의 데이터를 이용하여 DEPT 테이블에 추가하시오.

```
SQL> INSERT INTO dept
     SELECT  department_id, department_name  FROM    departments;
```

3. 기존 EMP 테이블을 삭제한 후 다음 테이블 인스턴스 차트를 기반으로 EMP 테이블을 생성하시오. 테이블을 생성한 후, 테이블이 생성되었는지 확인하시오.

열이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
키유형				
널/고유				
FK테이블				DEPT
FK 열				ID
데이터유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER
길이	7	25	25	7

```
SQL> DROP TABLE emp PURGE;
SQL> CREATE TABLE  emp
     (id          NUMBER(7),
      last_name     VARCHAR2(25),
      first_name    VARCHAR2(25),
      dept_id      NUMBER(7)CONSTRAINT emp_dept_id_fk REFERENCES dept(id));
SQL>  DESCRIBE emp
```

4. EMPLOYEES 테이블 구조를 기반으로 EMPLOYEES2 테이블을 생성하시오. EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPARTMENT_ID 열만 포함시키고 새 테이블의 열 이름을 각각 ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPT_ID로 지정하시오.

```
SQL> CREATE TABLE employees2
```

```
AS
SELECT  employee_id id, first_name, last_name, salary, department_id dept_id
FROM    employees;
```

5. EMPLOYEES2 테이블 상태를 읽기 전용으로 변경하시오.

```
SQL> ALTER TABLE employees2 READ ONLY
```

6. EMPLOYEES2 테이블에 다음 행을 삽입해 보시오.

```
SQL> INSERT INTO employees2
     VALUES (34, 'Grant','Marcie',5678,10)
```

→ "Update operation not allowed on table" 오류 메시지가 나타납니다. 테이블의 상태가 읽기 전용으로 지정되어 있으므로 행을 추가할 수 없습니다.

7. EMPLOYEES2 테이블을 읽기/쓰기 상태로 되돌리고 동일한 행을 다시 삽입해 보시오.

```
SQL> ALTER TABLE employees2 READ WRITE
SQL> INSERT INTO employees2
      VALUES (34, 'Grant','Marcie',5678,10)
```

8. EMPLOYEES2 테이블을 삭제하시오.

```
SQL> DROP TABLE employees2;
```


Practice 11

1. EMPLOYEES 테이블에서 사원 번호, 사원 이름 및 부서 번호를 기반으로 하는 EMPLOYEES_VU라는 뷰를 생성하시오. 사원 이름의 머리글을 EMPLOYEE로 변경하시오.

```
--권한 부족 오류 발생시 Run SQL Commandline(sqlplus) 실행 후 관리자로 접속하여 HR 사용자에게 CREATE VIEW, CREATE SYNONYM 권한을 부여한 후 실습 진행
SQL> CONN / as sysdba
SQL> GRANT create view, create synonym TO HR;
```

```
SQL> CREATE OR REPLACE VIEW employees_vu AS
     SELECT employee_id, last_name employee, department_id
     FROM employees;
```

2. EMPLOYEES_VU 뷰의 내용을 표시하시오.

```
SQL> SELECT   * FROM    employees_vu;
```

3. EMPLOYEES_VU 뷰를 사용하여 모든 사원의 이름 및 부서 번호를 표시하는 질의를 작성하시오.

```
SQL> SELECT   employee, department_id  FROM employees_vu;
```

4. 부서 50의 모든 사원에 대한 사원 번호, 사원 이름 및 부서 번호를 포함하는 DEPT50이라는 뷰를 생성하고 뷰의

열 레이블을 EMPNO, EMPLOYEE 및 DEPTNO로 지정하시오.

```
SQL> CREATE VIEW dept50 AS
     SELECT   employee_id empno, last_name employee, department_id deptno
     FROM     employees
     WHERE    department_id = 50;
```

5. DEPT50 뷰의 구조와 내용을 표시하시오.

```
SQL> DESCRIBE dept50
SQL> SELECT   * FROM     dept50;
```

6. DEPT50 뷰를 통해 Matos를 부서 80으로 변경해 보시오. DEPT50 뷰가 WITH CHECK OPTION 제약 조건을 사용하여 생성되었으므로 오류가 발생합니다. 이 제약 조건은 뷰에서 DEPTNO 열이 변경되지 않도록 합니다.

```
SQL> UPDATE   dept50
     SET      deptno = 80
     WHERE    employee = 'Matos';
```

7. 모든 사원의 이름, 부서 이름, 급여 및 급여 등급을 기반으로 하는 SALARY_VU라는 뷰를 생성하시오. EMPLOYEES, DEPARTMENTS 및 JOB_GRADES 테이블을 사용하고 열 레이블을 각각 Employee, Department, Salary 및 Grade로 지정하시오.

```
SQL> CREATE OR REPLACE VIEW salary_vu AS
     SELECT e.last_name "Employee", d.department_name "Department", e.salary "Salary",
          j.grade_level "Grades"
     FROM   employees e JOIN departments d
    ON  (e.department_id = d.department_id)
    JOIN job_grades j
     ON  (e.salary BETWEEN j.lowest_sal and j.highest_sal);
```


8. DEPT 테이블의 기본 키 열에 사용할 시퀀스를 생성하시오. 시퀀스 값은 300부터 시작하여 10씩 증가하며 최대 1000까지 가능하도록 하고 시퀀스 이름은 DEPT_ID_SEQ로 지정하시오. 기존 동일한 이름의 시퀀스가 존재하면 삭제를 먼저 하시오.

```
SQL> DROP SEQUENCE dept_id_seq;
SQL> CREATE SEQUENCE dept_id_seq
     START WITH 300
     INCREMENT BY 10
     MAXVALUE 1000;
```

9. DEPT 테이블에 두 행(row)을 다음과 같이 INSERT 하시오. ID 열에 대해서 생성한 시퀀스를 사용하시오. Education 및 Administration이라는 두 개의 부서를 추가하고 결과를 확인하시오.

```
SQL> INSERT INTO dept
     VALUES (dept_id_seq.nextval, 'Education');
```

```
SQL> INSERT INTO dept
     VALUES (dept_id_seq.nextval, 'Administration');
SQL> SELECT * FROM dept;
```

10. DEPT테이블의 NAME column에 비고유 인덱스를 생성하시오.

```
SQL> CREATE INDEX dept_name_idx ON dept(name)
```

11. EMPLOYEES 테이블에 대해 E 라는 동의어를 생성하시오.

```
SQL> CREATE SYNONYM e FOR employees;
```