Oracle Database : SQL FundamentalsⅡ

Practice 1

--이 단원의 실습은 개인 DB를 사용하는 학생의 경우에만 실습 가능합니다.

1. SQL Developer에서 관리자로 로그인하여 새로운 ROLE dev_1을 생성하고 다음의 권한을 부여하시오.

- CREATE TABLE
- CREATE SEQUENCE
- CREATE SYNONYM

```
SQL> CREATE ROLE dev_1;
SQL> GRANT create table, create sequence, create synonym TO dev_1;
```

2. ID와 Password가 DEV인 새 사용자를 생성하고 dev_1, connect ROLE과 CREATE VIEW 권한을 부여하시오.

```
SQL> CREATE USER dev IDENTIFIED BY dev;
SQL> GRANT connect TO dev;
SQL> GRANT create view TO dev;
```

3. SQL Developer에서 DEV 사용자 접속을 설정 후 DEV 사용자로 로그인하여 부여받은 권한을 조회하여 확인하시오.

```
SQL> SELECT * FROM session_privs;
```

4. 새로운 SQL Developer를 실행한 후 HR 사용자로 로그인(인사관리) 한다. 그런 다음 departments 테이블의 department_id, department_name, location_id 열을 조회할 수 있는 뷰를 dept_loc_vu 라는 이름으로 생성하고, HR.dept_loc_vu 뷰를 조회할 수 있는 객체권한을 DEV사용자에게 부여한다.

```
-- SQL Developer의 HR 세션(인사관리)
SQL> CREATE OR REPLACE VIEW dept_loc_vu
        AS
        SELECT department_id dept_id, department_name dept_name,
        location_id loc_id
        FROM departments;
SQL> GRANT select on hr.dept_loc_vu TO DEV;
-- SQL Developer의 DEV 세션
```

5. DEV 사용자 작업 창에서 hr.dept_loc_vu를 조회해 보고, SYNONYM을 "D" 라는 이름으로 생성한다.

```
SQL> SELECT * FROM dept_loc_vu; ➔ 결과는?
SQL> SELECT * FROM hr.dept_loc_vu;
SQL> CREATE SYNONYM d for hr.dept_loc_vu
SQL> SELECT * FROM d;
```

6. HR 사용자가 dept_loc_vu의 읽기 권한을 회수하거나 뷰를 삭제하면, DEV 사용자의 SYNONYM d는 어떻게 되는지 알아본다.

HR Session

SQL> REVOKE select on hr.dept_loc_vu FROM dev;

DEV Session

SQL> SELECT * FROM d;

HR Session

SQL> DROP VIEW dept_loc_vu;

DEV Session

SQL> SELECT object_name FROM user_objects; ➔ 결과는?

SQL> DROP SYNONYM d;

7. SQL Developer에서 HR 사용자로 로그인 한 다음 departments 테이블과 locations 테이블을 조인하여 department_id, department_name, city 열을 조회할 수 있는 뷰를 dept_loc_list_vu 라는 이름으로 생성하고, HR.dept_loc_list_vu 뷰를 조회할 수 있는 객체권한을 Public에게 부여하시오.

HR Session

```
SQL> CREATE OR REPLACE VIEW dept_loc_list_vu
        AS
        SELECT d.department_id dept_id, d.department_name dept_name, l.city
        FROM departments d JOIN locations l
        USING (location_id);
SQL> GRANT select on hr.dept_loc_list_vu TO public;
```

8. 관리자로 로그인하여 새로운 사용자 DEMO를 생성하고 Public Synonym DLIST를 hr.dept_loc_list_vu에 대하여 생성하시오.

```
SQL> CREATE USER demo IDENTIFIED BY demo;
SQL> GRANT connect TO demo;
SQL> CREATE PUBLIC SYNONYM  dlist FOR hr.dept_loc_list_vu;
```

9. dev, demo 등의 사용자로 로그인하여 Public Synonym DLIST를 사용해 보시오.

```
SQL> SELECT * FROM dlist;
```

10. 관리자로 로그인하여 DEMO 사용자의 암호를 oracle4u로 변경하고 dev 사용자는 삭제하시오.

```
SQL> alter user demo identified by oracle4u;
SQL> drop user dev cascade;
```

Practice 2

1. SQL Developer를 실행하고 HR 사용자로 로그인한 후 다음 테이블 인스턴트 차트를 기반으로 DEPT2 테이블을 생성하시오. 테이블을 생성한 후, 테이블이 생성되었는지 확인하시오. 실습 전에 기존의 EMP2, DEPT2가 있으면 삭제하시오.

```
SQL> DROP TABLE emp2 PURGE;
SQL> DROP TABLE dept2 PURGE;
```

열이름	ID	NAME
데이터유형	NUMBER	VARCHAR2
길이	7	25

```
SQL> CREATE TABLE dept2
        (id NUMBER(7),
         name VARCHAR2(25));
 SQL> DESCRIBE dept2
```

2. DEPARTMENTS 테이블의 데이터를 이용하여 DEPT2 테이블에 추가하시오. 필요한 열만 추가하시오.

```
SQL> INSERT INTO dept2
       SELECT   department_id, department_name
       FROM     departments;
```

3. 다음 테이블 인스턴스 차트를 기반으로 EMP2 테이블을 생성하시오.

열이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
데이터유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER
길이	7	25	25	7

```
SQL> CREATE TABLE   emp2
       (id              NUMBER(7),
        last_name       VARCHAR2(25),
        first_name      VARCHAR2(25),
        dept_id         NUMBER(7));
SQL> DESCRIBE emp2
```

4. 50바이트까지 긴 성을 가진 사원의 성을 표시할 수 있도록 EMP2 테이블을 수정한 후 수정 내용을 확인하시오.

```
SQL> ALTER TABLE emp2
       MODIFY (last_name    VARCHAR2(50));
SQL> DESCRIBE emp2
```

5. DEPT2 및 EMP2 테이블이 모두 데이터 딕셔너리에 저장되었는지 확인하시오(힌트: USER_TABLES).

```
SQL> SELECT    table_name FROM       user_tables
       WHERE     table_name IN ('DEPT2', 'EMP2');
```

6. 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;
```

7. EMP2 테이블을 삭제하시오.

```
SQL> DROP TABLE emp2;
```

8. recycle bin을 Query 하시오.

SQL> SELECT original_name, operation, droptime FROM recyclebin;

9. EMP2테이블을 Undrop 하시오.

SQL> FLASHBACK TABLE emp2 TO BEFORE DROP;

SQL> DESC emp2;

10. EMPLOYEES2 테이블에서 FIRST_NAME 열을 삭제한 후 DESCRIBE 명령으로 확인하시오.

SQL> ALTER TABLE employees2 DROP COLUMN FIRST_NAME;

SQL> DESCRIBE employees2

11. EMPLOYEES2 테이블의 DEPT_ID 열을 UNUSED로 표시한 후 DESCRIBE 명령으로 확인하시오.

SQL> ALTER TABLE employees2 SET UNUSED (dept_id);

SQL> DESCRIBE employees2

12. EMPLOYEES2 테이블에 있는 UNUSED 열을 모두 삭제하시오.

SQL> ALTER TABLE employees2 DROP UNUSED COLUMNS;

13. EMP2 테이블의 ID 열에 테이블 레벨의 PRIMARY KEY 제약 조건을 추가하시오. 제약 조건을 생성할 때 제약 조건의 이름을 my_emp_id_pk로 지정하시오.

SQL> ALTER TABLE emp2
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);

14. DEPT2 테이블의 ID 열에 PRIMARY KEY 제약 조건을 생성하시오. 제약 조건을 생성할 때 제약 조건의 이름을 my_dept_id_pk로 지정하시오.

SQL> ALTER TABLE dept2 ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(id);

15. EMP2 테이블에 존재하지 않는 부서에 사원이 배정되지 않도록 외래 키 참조를 EMP2 테이블에 추가하고 제약 조건의 이름을 my_emp_dept_id_fk로 지정하시오.

SQL> ALTER TABLE emp2
ADD CONSTRAINT my_emp_dept_id_fk FOREIGN KEY (dept_id) REFERENCES dept2(id);

16. USER_CONSTRAINTS 뷰를 질의하여 제약 조건이 추가되었는지 확인하고 제약 조건의 유형 및 이름을 적어두시오.

SQL> SELECT constraint_name, constraint_type FROM user_constraints
WHERE table_name IN ('EMP2', 'DEPT2');

17. USER_OBJECTS 데이터 딕셔너리 뷰에서 EMP2 및 DEPT2 테이블의 객체 이름 및 유형을 표시하시오. 새 테이블 및 새 인덱스가 생성된 것을 볼 수 있습니다. 인덱스의 이름은 제약조건의 이름과 동일합니다.

SQL> col object_name format a30

SQL> SELECT object_name, object_type FROM user_objects
WHERE object_name IN ('EMP2','DEPT2')

18. EMP2 테이블을 수정하여 십진 자릿수 2, 소수점 이하 자릿수 2인 NUMBER 데이터 유형의 COMMISSION 열을 추가하시오. 커미션 값이 0보다 크도록 커미션 열에 제약 조건을 추가하시오.

```
SQL> ALTER TABLE EMP2
        ADD commission NUMBER(2,2) CONSTRAINT my_emp_comm_ck CHECK (commission >= 0) ;
```

19. EMP2, DEPT2 테이블을 영구히 삭제하고 recycle bin을 확인하시오.

```
SQL> DROP TABLE emp2 PURGE
SQL> DROP TABLE dept2 PURGE;
SQL> SHOW recyclebin;
```

20. 다음 테이블 인스턴스 차트를 기반으로 DEPT_NAMED_INDEX 테이블을 생성하시오. Primary Key에 대한 Index의 이름을 DEPTNO_IDX로 지정하시오.

열이름	Deptno	Dname
Primary Key	Yes	
데이터유형	NUMBER	VARCHAR2
길이	4	30

```
SQL> CREATE TABLE DEPT_NAMED_INDEX
        (deptno NUMBER(4) PRIMARY KEY USING INDEX
                                (CREATE INDEX depno_idx ON DEPT_NAMED_INDEX(deptno)),
         Dname varchar2(30));
```

21. USER_INDEXES 테이블을 질의하여 DEPT_NAMED_INDEX 테이블의 INDEX_NAME을 표시하시오.

```
SQL> SELECT INDEX_NAME, TABLE_NAME FROM USER_INDEXES
        WHERE TABLE_NAME = 'DEPTNO_INDEX';
```

Practice 3

1. 지정된 테이블에 대하여, 열 이름, 데이터 타입, 길이 Null 허용 여부를 질의를 실행하시오.

```
SQL> SELECT column_name, data_type, data_length, nullable FROM user_tab_columns
        WHERE table_name='EMPLOYEES';
```

2. 지정된 테이블에 대한 USER_CONSTRAINTS와 USER_CONS_COLUMNS를 조인하여 열 이름, 제약조건 이름, 제약조건 type, search_condition 및 상태를 조회하시오.

```
SQL> SELECT ucc.column_name, uc.constraint_name, uc.constraint_type,
                uc.search_condition, uc.status
        FROM user_constraints uc JOIN user_cons_columns ucc
        ON uc.table_name =ucc.table_name
        AND uc.constraint_name=ucc.constraint_name
        AND uc.table_name='EMPLOYEES';
```

3. DEPARTMENTS 테이블 정의에 테이블을 설명하는 주석을 추가한 후 데이터 딕셔너리에서 추가한 항목을 확인하시오.

```
SQL> COMMENT ON TABLE departments IS 'Department Information';
SQL> SELECT   * FROM      user_tab_comments
       WHERE    table_name = 'DEPARTMENTS'
```

4. HR Schema의 모든 동의어의 이름을 질의하시오.

```
SQL> SELECT * FROM user_synonyms;
```

5. USER_VIEWS 데이터 딕셔너리 뷰에서 뷰 이름 및 텍스트를 선택하시오.

```
SQL> SET LONG 600
SQL> SELECT    view_name, text   FROM   user_views;
```

6. 시퀀스 이름, 최대값, 증가분, 마지막 번호와 같은 시퀀스 정보를 표시하는 질의를 작성하시오.

```
SQL> SELECT    sequence_name, max_value, increment_by, last_number
       FROM       user_sequences;
```

7. HR 소유의 모든 객체의 이름과 유형을 표시하는 질의를 작성하시오.

```
SQL> SELECT object_name, object_type FROM user_objects;
```

8. SALES_DEPT 테이블을 다음 사양으로 만드시오. PRIMARY KEY 열에 대한 인덱스의 이름을 SALES_PK_IDX로 지정한 후 데이터 딕셔너리로부터 인덱스이름, 테이블이름 그리고 고유 인덱스 인지 여부를 확인하시오.

```
SQL> CREATE TABLE sales_dept
       (team_id NUMBER(3) PRIMARY KEY USING INDEX
        (CREATE INDEX sales_pk_idx ON SALES_DEPT(team_id)),
        Location VARCHAR2(30));
SQL> SELECT index_name, table_name, uniqueness FROM user_indexes
       WHERE table_name = 'SALES_DEPT';
```

Practice 4

1. Multi Table Insert 기능을 사용해 보시오.

a) 실습에 필요한 테이블 생성을 위해 스크립트를 실행하시오.

```
SQL> @c:₩db_test₩sql_labs₩cre_minstab.sql
SQL> SELECT * FROM tab;
```

b) Unconditional All Insert 구문을 사용하여 선택된 행이 INTO 절에 지정된 모든 테이블에 입력되는 것을 확인하시오.

```
SQL> INSERT   ALL
INTO sal_history VALUES(EMPID,HIREDATE,SAL)
INTO mgr_history VALUES(EMPID,MGR,SAL)
SELECT employee_id EMPID, hire_date HIREDATE, salary SAL, manager_id MGR
FROM   employees
```

```
WHERE employee_id > 200;
SQL> SELECT * FROM sal_history;
SQL> SELECT * FROM mgr_history;
SQL> rollback;
```

c) Conditional All Insert 구문을 활용하여 선택된 행이 WHEN절의 조건에 따라 INTO 테이블에 선택적으로 저장되는 것을 확인하시오.

```
SQL> INSERT ALL
        WHEN SAL > 10000 THEN
        INTO sal_history VALUES(EMPID,HIREDATE,SAL)
        WHEN MGR > 200     THEN
        INTO mgr_history VALUES(EMPID,MGR,SAL)
        SELECT employee_id EMPID,hire_date HIREDATE,    salary SAL, manager_id MGR
         FROM     employees
        WHERE    employee_id > 200;
```

d) Conditional First Insert 구문을 사용해 보시오. 선택된 행가운데 첫번째 WHEN 절을 만족하는 데이터는 첫번째 INTO 테이블에만 저장됩니다. 나머지 조건에 대해서는 Contitional All Insert 방식으로 행이 입력됩니다.

```
SQL> INSERT FIRST
        WHEN SAL   > 25000   THEN
       INTO special_sal VALUES(DEPTID, SAL)
       WHEN HIREDATE like ('%00%') THEN
       INTO hiredate_history_00 VALUES(DEPTID,HIREDATE)
       WHEN HIREDATE like ('%99%') THEN
       INTO hiredate_history_99 VALUES(DEPTID, HIREDATE)
       ELSE   INTO hiredate_history VALUES(DEPTID, HIREDATE)
       SELECT department_id DEPTID, SUM(salary) SAL, MAX(hire_date) HIREDATE
       FROM     employees
       GROUP BY department_id;
```

e) 비관계형 데이터베이스 테이블의 각 레코드를 관계형 데이터베이스 테이블의 다중 레코드로 변환하는 등의 변형 작업을 수행하는 것을 Pivot이라고 합니다. 비관계형 데이터베이스 테이블 SALES_SOURCE_DATA의 각 레코드가 SALES_INFO 테이블의 다섯 개의 레코드로 변환되도록 Pivot Insert 구문을 작성하시오.

```
SQL> SELECT * FROM sales_source_data;
SQL> DESC sales_info
SQL> SELECT * FROM sales_info;
SQL> INSERT ALL
        INTO sales_info VALUES (employee_id,week_id,sales_MON)
        INTO sales_info VALUES (employee_id,week_id,sales_TUE)
        INTO sales_info VALUES (employee_id,week_id,sales_WED)
```

```
        INTO sales_info VALUES (employee_id,week_id,sales_THUR)
        INTO sales_info VALUES (employee_id,week_id, sales_FRI)
        SELECT EMPLOYEE_ID, week_id, sales_MON, sales_TUE,
                sales_WED, sales_THUR,sales_FRI
        FROM sales_source_data;
SQL> SELECT * FROM sales_info;
```

2. Merge문을 실습해 봅니다.

a) 실습에 필요한 EMP13 테이블을 생성합니다. 테이블 생성 후 EMP13 테이블의 데이터를 확인하시오.

```
SQL> @c:₩db_test₩sql_labs₩cre_emp13.sql
```

b) EMLOYEES 테이블의 내용을 EMP13 테이블에 병합하시오. 병합 시 사원번호를 사용하여 동일한 행이 발견되면 EMPLOYEES 내용을 EMP13에 적용하기 위하여 UPDATE 및 INSERT가 되어야하고 동일한 행이 발견되지 않으면 새 행으로 처리되어 EMP13에 삽입됩니다.

```
SQL> MERGE INTO emp13 c
       USING employees e
       ON (c.employee_id = e.employee_id)
    WHEN MATCHED THEN
       UPDATE SET
         c.last_name         = e.last_name,
         c.job_id            = e.job_id,
         c.salary            = e.salary,
         c.department_id   = e.department_id
    WHEN NOT MATCHED THEN
       INSERT VALUES(e.employee_id, e.last_name,e.job_id,
            e.salary, e.department_id);
SQL> COMMIT;
```

c) EMP13테이블을 조회하여 처음 세 행의 UPDATE 및 나머지 행의 INSERT를 확인하시오.

```
SQL> SELECT * FROM emp13;
```

3 .Flashback Query 기술을 사용해 봅니다.

a) 실습에 필요한 테이블을 생성합니다.

```
SQL> CREATE TABLE dept3
        AS SELECT * FROM departments;
SQL> SELECT * FROM dept3;
```

b) DEPT3 테이블의 한 행을 수정합니다.

```
SQL> UPDATE dept3
  SET department_name = 'Admin'
  WHERE department_id = 80;
```

```
SQL> COMMIT;
SQL> SELECT * FROM dept3;
```

c) 5분 정도 흐른 뒤, 현재 시각으로부터 5분전의 DEPT3 테이블의 상태를 다음 명령문으로 확인합니다.

```
SQL> SELECT * FROM dept3
       AS OF TIMESTAMP (SYSTIMESTAMP - 5/1440);
```

d) 최근 80번 부서의 이름을 실수로 변경했다면, HR.DEPT3 테이블을 이전 상태로 되돌리기 위해서는 다음과 같은 서브 쿼리를 사용하여 UPDATE를 실행할 수 있습니다.

```
SQL> UPDATE dept3
      SET department_name = (SELECT department_name FROM dept3
                             AS OF TIMESTAMP (SYSTIMESTAMP-7/1440)
                              WHERE department_id = 80)
      WHERE department_id =80;
SQL> COMMIT;
SQL> SELECT * FROM dept3;
```

2. Flashback Versions Query를 실행해 봅니다.

a) EMP3 테이블을 생성하고 동일한 행을 여러 번 수정합니다.

```
SQL> CREATE TABLE emp3
      AS SELECT employee_id empno, last_name ename, salary sal FROM employees;
SQL> SELECT * FROM emp3
       WHERE empno = 101;
SQL> UPDATE emp3
  SET sal = 17000
  WHERE empno = 101 ;
SQL> COMMIT;
SQL> UPDATE emp3
set sal = 15000
  WHERE empno = 101 ;
SQL> COMMIT;
SQL> DELETE FROM emp3
      WHERE empno = 101;
SQL> COMMIT;
```

b) EMP3 테이블의 현재 상태를 봅니다.

```
SQL> SELECT * FROM emp3
       WHERE empno = 101;
```

c) 다음 쿼리는 101 사원의 변경 내용을 나타냅니다.

```
SQL> col versions_starttime for a30
```

```
SQL> col versions_endtime for a30
SQL> set linesize 120
SQL> SELECT   versions_starttime, versions_endtime, versions_xid, versions_operation, sal
       FROM emp3 VERSIONS BETWEEN TIMESTAMP minvalue AND maxvalue
       WHERE empno = 101;
```

Practice 5

1. NLS_DATE_FORMAT을 DD-MON-YYYY HH24:MI:SS로 설정하여 세션을 변경하시오.

```
SQL>   ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/MM/DD HH24:MI:SS';
```

2. 다음 실습을 수행하시오

a) 다음 시간대에 대해 시간대 오프셋(TZ_OFFSET)을 표시하는 질의를 작성하시오.

```
SQL> SELECT TZ_OFFSET ('US/Pacific-New') from dual;
SQL> SELECT TZ_OFFSET ('Singapore') from dual;
SQL> SELECT TZ_OFFSET ('Egypt') from dual;
```

b)TIME_ZONE 파라미터 값을 US/Pacific-New의 시간대 오프셋으로 설정하여 세션을 변경하시오.

```
SQL> ALTER SESSION SET TIME_ZONE = '-7:00';
```

c) 이 세션에 대한 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시하시오. (참고: 명령을 실행하는 날짜에 따라 결과가 다를 수 있습니다.)

```
SQL> SELECT CURRENT_DATE, CURRENT_TIMESTAMP, LOCALTIMESTAMP
       FROM DUAL;
```

d) TIME_ZONE 파라미터 값을 Singapore의 시간대 오프셋으로 설정하여 세션을 변경하시오.

```
SQL> ALTER SESSION SET TIME_ZONE = '+8:00';
```

e) 이 세션에 대한 CURRENT_DATE, CURRENT_TIMESTAMP 및 LOCALTIMESTAMP를 표시하시오.

```
SQL>   SELECT CURRENT_DATE, CURRENT_TIMESTAMP, LOCALTIMESTAMP
        FROM DUAL;
```

3. 부서 ID가 80인 사원에 대해 EMPLOYEES 테이블의 HIRE_DATE 열에서 YEAR를 추출하는 질의를 작성하시오.

```
SQL> SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)
       FROM employees
       WHERE department_id = 80;
```

4. NLS_DATE_FORMAT을 DD-MON-YYYY로 설정하여 세션을 변경하시오.

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/MM/DD';
```

5. 다음 실습을 하시오.

a) SAMPLE_DATES 테이블을 생성하고 조회하시오.

```
SQL> CREATE TABLE sample_dates (sales_date date);
SQL> Insert into sample_dates values (sysdate);
SQL> SELECT * FROM sample_dates;
```

b) SALES_DATE 컬럼의 데이터타입을 timestamp 타입으로 변경하고 데이터를 조회하시오.

```
SQL> ALTER TABLE sample_dates MODIFY sales_date TIMESTAMP;
SQL> SELECT * FROM sample_dates;
```

6. EMPLOYEES 테이블에서 last_name과 다음을 만족하는 Review 를 질의하여 표시합니다.

- 1998년에 입사한 사원은 'Needs Review'로 표시
- 나머지는 모두 'Not this year!'로 표시

```
SQL> SELECT e.last_name,
        (CASE extract(year from e.hire_date)
            WHEN 1998 THEN 'Needs Review'
            ELSE 'not this year!'
            END )                AS "Review "
    FROM    employees e
    ORDER BY e.hire_date;
```

Practice 6

1. 실습을 위해 empl_demo 테이블을 생성한 후, 커미션을 받는 사원과 부서 번호 및 급여가 일치하는 사원의 이름, 부서 번호 및 급여를 표시하는 질의를 작성하시오.

```
SQL> @c:₩db_test₩sql_labs₩cre_empl.sql
SQL> SELECT last_name, department_id, salary
       FROM    empl_demo
       WHERE   (salary, department_id) IN
                (SELECT   salary, department_id FROM   empl_demo
                 WHERE    commission_pct IS NOT NULL);
```

2. 위치 ID가 1700인 사원과 급여 및 커미션이 일치하는 사원의 이름, 부서 이름 및 급여를 표시하시오.

```
SQL>   SELECT e.last_name, d.department_name, e.salary
       FROM    empl_demo e JOIN departments d
       ON   (e.department_id = d.department_id)
       WHERE    (salary, NVL(commission_pct,0)) IN
                 (SELECT salary, NVL(commission_pct,0)
                  FROM    empl_demo   e JOIN departments   d
                   ON      (e.department_id = d.department_id)
                   AND     d.location_id = 1700);
```

3. Kochhar와 동일한 급여 및 커미션을 받는 모든 사원의 이름, 입사일 및 급여를 표시하는 질의를 작성하시오. 단, 결과 집합에 Kochhar는 표시하지 않습니다.

```
SQL> SELECT last_name, hire_date, salary FROM empl_demo
        WHERE   (salary, NVL(commission_pct,0)) IN
                    (SELECT salary, NVL(commission_pct,0) FROM empl_demo
                     WHERE   last_name = 'Kochhar')
                      AND last_name != 'Kochhar';
```

4. 모든 영업 관리자(JOB_ID = 'SA_MAN')보다 급여를 많이 받는 사원을 표시하는 질의를 작성하고 결과를 최고 급여에서 최저 급여의 순으로 정렬하시오.

```
SQL> SELECT last_name, job_id, salary FROM     employees
      WHERE   salary > ALL (SELECT salary FROM employees
                                WHERE   job_id = 'SA_MAN')
      ORDER BY salary DESC;
```

5. 문자 *T*로 시작하는 도시에 사는 사원의 세부 정보인 사원 ID, 이름 및 부서 ID를 표시하시오.

```
SQL> SELECT employee_id, last_name, department_id FROM     employees
      WHERE   department_id IN (SELECT department_id FROM departments
                                    WHERE location_id IN (SELECT   location_id FROM locations
                                                          WHERE city LIKE 'T%'));
```

6. 관리자가 아닌 사원을 모두 찾으시오.

a) 먼저 NOT EXISTS 연산자를 사용하여 이 작업을 수행하시오.

```
SQL> SELECT outer.last_name FROM       employees outer
      WHERE   NOT EXISTS (SELECT 'X' FROM employees inner
                              WHERE inner.manager_id = outer.employee_id);
```

b) NOT IN 연산자를 사용하여 이 작업을 수행해 보고 두 방법을 비교해 봅니다.

```
SQL> SELECT outer.last_name FROM     employees outer
       WHERE   outer.employee_id   NOT IN (SELECT inner.manager_id FROM     employees inner);
```

Note) 이 방법은 좋은 방법이 아닙니다. 서브 쿼리에서 NULL 값을 선택하므로 전체 질의에서 아무런 행도 반환하지 않습니다. 그 이유는 NULL 값을 비교하는 조건에서는 결과가 항상 NULL이기 때문입니다. 값 집합에 NULL 값이 포함될 가능성이 있을 경우에는 NOT EXISTS 대신 NOT IN을 사용하지 않도록 합니다.

7. 소속 부서의 평균 급여보다 적은 급여를 받는 사원의 이름을 표시하는 질의를 작성하시오.

```
SQL> SELECT last_name FROM       employees outer
      WHERE outer.salary < (SELECT AVG(inner.salary) FROM employees inner
                                WHERE inner.department_id   = outer.department_id);
```

8. 소속 부서에서 입사일이 늦지만 더 많은 급여를 받는 동료가 있는 사원의 이름을 표시하는 질의를 작성하시오.

```
SQL> SELECT last_name FROM   employees outer
       WHERE EXISTS (SELECT 'X' FROM employees inner
                           WHERE inner.department_id = outer.department_id
                           AND inner.hire_date > outer.hire_date
                           AND inner.salary > outer.salary);
```

9. 모든 사원의 사원 ID, 이름 및 부서 이름을 표시하는 질의를 작성하시오. 스칼라 서브 쿼리를 사용하여 SELECT 문에서 부서 이름을 검색해서 해결할 수 있습니다.

```
SQL> SELECT employee_id, last_name,
                    (SELECT department_name FROM departments d
                     WHERE     e.department_id = d.department_id ) department
       FROM employees e
       ORDER BY department;
```

10. 총 급여가 전체 회사 총 급여의 1/8보다 많은 부서의 이름을 표시하는 질의를 작성하시오. 이 질의를 WITH 절을 사용하여 작성하되, SUMMARY라는 이름을 부여하시오.

```
SQL> WITH
      summary AS (
      SELECT d.department_name, SUM(e.salary) AS dept_total
      FROM employees e, departments d
      WHERE e.department_id = d.department_id
      GROUP BY d.department_name)
      SELECT department_name, dept_total
      FROM summary
      WHERE dept_total > ( SELECT SUM(dept_total) * 1/8 FROM summary   )
      ORDER BY dept_total DESC;
```

Practice 7

1. 스크립트를 실행하여 실습에 필요한 T1 테이블을 생성하시오.

```
SQL> @/home/oracle/labs/regexp_tab.sql
SQL> SELECT * FROMt1;
```

2. 정규표현식을 사용하여 데이터를 검색해보시오.

a) 'Steven' 또는 'Stephen' 의 문자열을 검색하시오.

```
SQL> SELECT fname, lname
       FROM t1
       WHERE REGEXP_LIKE (fname, '^Ste(v|ph)en$') ;
```

b) 전화번호가 3자리.2자리.4자리.6자리 구조인 번호를 검색하시오. 임의의 한 문자를 비교 하기 때문에 검색 결과 중 문자 ABC 가 포함 된 것도 함께 검색 된다.

```
SQL> SELECT fname, phone
        FROM t1
        WHERE REGEXP_LIKE (phone, '...₩...₩.....₩.......') ;
```

c) 다음과 같이 각각의 자리마다 숫자로 반복 되는 회수를 지정할 수 있으며 원하는 문자열이 포함 된 것을 찾을 수 있다.

```
SQL> SELECT fname, phone
      FROM t1
      WHERE REGEXP_LIKE (phone, '₩d{3}₩.₩d{2}₩.₩d{4}₩.₩d{6}') ;
```

d) 정규식에 의한 패턴을 찾아 "." 으로 구분 된 문자를 "-" 으로 변경, 즉 대체문자열로 변경을 할 수 있다.

```
SQL> SELECT fname, phone, REGEXP_REPLACE ( phone , '₩.' , '-' ) new_format
      FROM t1 ;
```

e) 정규식 패턴 비교를 통해 위치 값을 확인 할 수 있으며, 다음 예제는 지정된 Class 가 알파벳을 찾는 부분이므로 첫 번째 알파벳 문자의 위치를 검색 한다.

```
SQL> SELECT fname, addr, REGEXP_INSTR ( addr, '[[:alpha:]]' ) pos , phone,
            REGEXP_INSTR ( phone, '[[:alpha:]]') pos
      FROM t1;
```

f) 정규식 패턴을 검색하여 부분 문자를 추출할 수 있으며, 다음 예제는 정규식을 사용하여 두 번째 문자열 (Road) 추출한다.

```
SQL> SELECT fname, addr, REGEXP_SUBSTR ( addr, ' [^ ]+ ') road
      FROM t1 ;
```

g) 정규식 패턴을 검색하여 발견 된 횟수를 계산 할 수 있다. 주소에서 'a' 가 발견된 횟수를 검색한다.

```
SQL> SELECT fname, addr, REGEXP_COUNT(addr,'a') cnt
      FROM t1;
```