```
REM SUBPROGRAM
--1. 잘 정리된 논리적 코드분할이름
--2. 컴파일된 상태에서 데이터베이스에 저장되기 때문에 성능이 향상된다.
--3. 테이블이름이나 컬럼의 이름을 명시하지 않기 때문에 보안에 도움이 된다.
--4. 모듈화를 통한 관리 용이
--5. 종류
  --1) Stored Procedure   <-----Java 에서 사용
  --2) Stored Function

REM Stored PROCEDURE(저장프로시저)
--1. 목적 : 속도, 보안
--2. compile(pcode로 변환)상태로 RDBMS 에 저장
--3. 나중에 실행될 일련의 명령의 집합
--4. 리턴하는 행의 갯수가 많으면 에러난다.
--5. PL/SQL에서 SELECT는 반드시 SELECT INTO 를 사용해야
--6. Syntax
    CREATE OR REPLACE PROCEDURE procedure_name
    (
        매개변수 영역
        변수이름     변수모드    변수타입
        v_empno      IN(OUT, IN OUT)        emp.EMPNO%TYPE
    )
    IS
        내부변수
    BEGIN
       PL/SQL 문장들
    END;

    --OR REPLACE :이미 존재할 경우 procedure 의 내용을 지우고 다시 생성
    --변수 Mode  : 3가지
       --IN : 입력 매개변수
       --OUT : 출력 매개변수
       --IN OUT : 입력, 출력 매개변수

--7. Guide Lines
  --1) 어떤 parameter도 가능
  --2) IS 로 시작
  --3) local 변수 선언은 IS와 BEGIN 사이에 선언
  --4) PL/SQL 에서 SELECT시 반드시 INTO 사용
  --5) SELECT시 여러 개의 행을 리턴한다면 에러 발생

    CREATE OR REPLACE PROCEDURE test proc
    IS
        v_name   VARCHAR2(30);
    BEGIN
        v name := '한지민';
        DBMS_OUTPUT.PUT_LINE('My name is ' || v_name);
    END;
    /

    EXEC test_proc;

    SET SERVEROUTPUT ON
    CREATE OR REPLACE PROCEDURE sp_HelloWorld
    IS
    BEGIN
        DBMS OUTPUT.PUT LINE('Hello, World');
    END;
    /

    EXEC sp HelloWorld;

-- emp 테이블이 모든 데이터를 삭제하는 Stored Procedure 를 작성하시오.
    CREATE OR REPLACE PROCEDURE del all
    IS
    BEGIN
          DELETE FROM emp;
    END;
    /

    EXEC del all

--8. 오류 발생시 오류 보기
    DESC USER_ERRORS
    SHOW ERROR

--9. 확인하기
    DESC USER_PROCEDURES;
    SELECT OBJECT NAME, PROCEDURE NAME FROM USER PROCEDURES;
    DESC USER_SOURCE;
    COL NAME FOR A10
    COL TEXT FOR A40
    SELECT NAME, TEXT FROM USER SOURCE;

--10. IN 매개변수
    CREATE OR REPLACE PROCEDURE test proc
    (
        p_name  IN VARCHAR2
    )
    IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('My name is ' || p_name);
    END;
    /
```

```
    EXEC test proc('설운도');

    SET SERVEROUTPUT ON
    CREATE OR REPLACE PROCEDURE sp_OneInParameter
    (param1   IN    VARCHAR2)
    IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Hello, World IN parameter ' || param1);
    END;
    /

    EXEC sp OneInParameter('Michael');
    Hello, World IN parameter Michael

--사원번호와 봉급을 입력받아 업데이트하는 PL/SQL 문장을 완성하시오.

    SET SERVEROUTPUT ON
    ACCEPT p_empno  PROMPT   'Enter a Number : '
    ACCEPT p_sal       PROMPT   'Enter a Salary : '
    DECLARE
        v_empno      emp.empno%TYPE  := &p_empno;
        v_sal           emp.sal%TYPE  := &p_sal;
    BEGIN
        UPDATE emp
        SET sal = v_sal
        WHERE empno = v_empno;
        COMMIT;
    END;
    /
    SET SERVEROUTPUT OFF


    CREATE OR REPLACE PROCEDURE emp_sal_update
    (
        v empno    IN  emp.empno%TYPE,
        v_sal    IN emp.sal%TYPE
    )
    IS
    BEGIN
        UPDATE emp
        SET sal = v_sal
        WHERE empno = v empno;
        COMMIT;
    END;
    /

    -- 실행 EXEC[UTE] Procedure_name(parameter1, parameter2,...);
    EXEC emp_sal_update(7788, 8000);

----사번을 받아 삭제하는 프로시저
    CREATE OR REPLACE PROCEDURE emp_del
    (
         v empno     IN     emp.empno%TYPE
    )
    IS
    BEGIN
        DELETE FROM emp
        WHERE empno = v_empno;
    END;
    /
    --실행할 때
    EXEC emp_del(7900);

--부서번호, 부서이름, 지역을 받아 삽입하는 프로시저
    CREATE OR REPLACE PROCEDURE sp_insert_dept
    (
        v_deptno    IN     dept.deptno%TYPE,
        v dname     IN     dept.dname%TYPE,
        v_loc       IN     dept.loc%TYPE
    )
    IS
    BEGIN
        INSERT INTO DEPT
        VALUES (v_deptno, UPPER(v_dname), UPPER(v_loc));
    END;
    /

    EXEC sp_insert_dept(50, 'marketting', 'yatap');

--emp table에서 새로운 사원의 정보를 이름, 업무, 매니저, 급여를 입력받아 등록하는 emp_input 프로시저를
생성하여라. 단, 부서번호는 매니저의 부서 번호와 동일하게 하고 보너스는 SALESMAN은 0을 그 외는 NULL을 입력하라.

    CREATE SEQUENCE emp empno seq
        START WITH 1
        INCREMENT BY 1
        MAXVALUE  999
        NOCYCLE
        CACHE 10;

    CREATE PROCEDURE emp input
    (
        v_name  IN      emp.ename%TYPE,
        v_job   IN      emp.job%TYPE,
        v mgr   IN      emp.mgr%TYPE,
        v_sal   IN      emp.sal%TYPE
```

```
)
IS
    v_comm  emp.comm%TYPE;
    v_deptno        emp.deptno%TYPE;
BEGIN
    IF UPPER(v_job) = 'SALESMAN' THEN
        v_comm := 0;
    ELSE
        v comm := NULL;
    END IF;

    SELECT deptno
    INTO    v deptno
    FROM emp
    WHERE empno = v_mgr;

    INSERT INTO emp
    VALUES(emp_empno_seq.NEXTVAL, v_name, UPPER(v_job), v_mgr, SYSDATE, v_sal, v_comm, v_deptno);
END;
/

--11. OUT 매개변수
CREATE OR REPLACE PROCEDURE test proc
(
    p_name  OUT VARCHAR2
)
IS
BEGIN
    p_name := 'My name is 한지민.';
END;
/

--실행
--OUT 파라미터는 Binding 변수가 필요하다.
--VAR[IABLE]  변수이름  변수타입
--OUT 변수값 출력
--PRINT 변수이름
VAR g ename VARCHAR2(20);
EXEC test proc(:g ename);    --binding 변수
PRINT g_ename;

    OR

DECLARE
    v_name    VARCHAR2(100);
BEGIN
    test_proc(v_name);
    DBMS_OUTPUT.PUT_LINE(v_name);
END;
/

-----------------------------------------------
SET SERVEROUTPUT ON
CREATE OR REPLACE PROCEDURE sp_OneOutParameter
(outParam1   OUT    VARCHAR2)
IS
BEGIN
    outParam1 := 'Hello World OUT parameter';
END;
/

SQL> variable a VARCHAR2(30);
SQL> EXEC sp_OneOutParameter(:a);

PL/SQL procedure successfully completed.

SQL> print a;

A
--------------------------------
Hello World OUT parameter

OR

SET SERVEROUTPUT ON
DECLARE
    outParam1   VARCHAR2(100);
BEGIN
    sp OneOutParameter(outParam1);
    DBMS_OUTPUT.PUT_LINE(outParam1);
END;
/

SQL> @demo.sql
Hello World OUT parameter

PL/SQL procedure successfully completed.

OR

--굳이 Stored Procedure를 별도로 생성할 필요가 없을때는
SET SERVEROUTPUT ON
DECLARE
  outParam1 VARCHAR2(100);
```

```
    PROCEDURE procOneOUTParameter
    (outParam1 OUT VARCHAR2)
    IS
    BEGIN
      outParam1 := 'Hello World OUT parameter';
    END;

    BEGIN
      procOneOUTParameter(outParam1);
      DBMS_OUTPUT.PUT_LINE(outParam1);
    END;
    /

    -------------------------------------------------
    SEE SERVEROUTPUT ON
    DECLARE
        a  NUMBER;
        b  NUMBER;
        c  NUMBER;

    PROCEDURE findMin(x IN number, y IN number, z OUT NUMBER) IS
    BEGIN
        IF x < y THEN
            z := x;
        ELSE
            z := y;
        END IF;
    END;

    BEGIN
        a := 23;
        b := 45;
        findMin(a, b, c);
        DBMS OUTPUT.PUT LINE(' Minimum of (23, 45) : ' || c);
    END;
    /

    SQL> @demo.sql
    Minimum of (23, 45) : 23

    PL/SQL procedure successfully completed.

    ----------------------------------------------------------------

--사번을 받아 사원이름과 봉급 검색
    CREATE OR REPLACE PROCEDURE sp emp select
    (
        v_empno IN  emp.EMPNO%TYPE,
        v ename OUT emp.ENAME%TYPE,
        v sal   OUT emp.sal%TYPE
    )
    IS
    BEGIN
          SELECT ename, sal
          INTO v_ename, v_sal
          FROM EMP
          WHERE empno = v empno;
    END;
    /

    SQL> var g ename VARCHAR2(20);
    SQL> var g_sal    NUMBER;
    SQL> EXEC sp_emp_select(7566, :g_ename, :g_sal);

    PL/SQL 처리가 정상적으로 완료되었습니다.

    SQL> print g_ename;

    G_ENAME
    --------------------------------
    JONES

    SQL> print g_sal;

    G SAL
    ----------
    2975

--이름을 입력받아서 그 사원의 정보 중 부서명과 급여를 검색하는 프로시저를 완성하시오.
    CREATE OR REPLACE PROCEDURE emp_dept_sal_select
    (
        v_ename        IN       emp.ename%TYPE,
        v dname        OUT     dept.dname%TYPE,
        v_sal             OUT     emp.sal%TYPE
    )
    IS
        v deptno      emp.deptno%TYPE;
    BEGIN
        SELECT deptno, sal
        INTO   v deptno, v sal
        FROM emp
        WHERE ename = UPPER(v_ename);

        SELECT  dname
        INTO v_dname
```

```
        FROM dept
        WHERE deptno = v deptno;
    END;
    /

    VAR g_dname VARCHAR2(20);
    VAR g_sal NUMBER;
    EXEC emp_dept_sal_select('scott', :g_dname, :g_sal);

    PL/SQL PROCEDURE successfully completed.

    PRINT g dname;
    PRINT g sal;

    OR

    SET SERVEROUTPUT ON
    DECLARE
        v_ename    emp.ename%TYPE;
        v dname    dept.dname%TYPE;
        v_sal         emp.sal%TYPE;
    BEGIN
        v ename := 'scott';
        emp dept sal select(v ename, v dname, v sal);
        DBMS_OUTPUT.PUT_LINE('Name : ' || v_ename);
        DBMS_OUTPUT.PUT_LINE('Dept Name : ' || v_dname);
        DBMS_OUTPUT.PUT_LINE('Salary : ' || v sal);
    END;
    /

    SQL> @demo.sql
    Name : scott
    Dept Name : RESEARCH
    Salary : 3000

    PL/SQL procedure successfully completed.

-------------------------------------------------------------------
--우편번호검색, 단 동이름은 역삼동으로만 할 것
    CREATE OR REPLACE PROCEDURE sp_zipcode_select
    (
        v dong1  IN zipcode.DONG%TYPE,
        v_zipcode    OUT    zipcode.ZIPCODE%TYPE,
        v_sido   OUT    zipcode.SIDO%TYPE,
        v_gugun  OUT    zipcode.GUGUN%TYPE,
        v dong   OUT    zipcode.DONG%TYPE,
        v_bunji  OUT    zipcode.BUNJI%TYPE
    )
    IS
    BEGIN
        SELECT zipcode.ZIPCODE, sido, gugun, dong, bunji
        INTO v_zipcode, v_sido, v_gugun, v_dong, v_bunji
        FROM ZIPCODE
        WHERE dong LIKE CONCAT(CONCAT('%', v_dong1),'%');
    END;
    /

--12. IN OUT 파라미터
    CREATE OR REPLACE PROCEDURE test_proc
    (
        p name  IN OUT VARCHAR2
    )
    IS
    BEGIN
        p_name := 'My name is ' || p_name;
    END;
    /

    DECLARE
        v_name   VARCHAR2(100);
    BEGIN
        v name := '나훈아';
        test_proc(v_name);
        DBMS_OUTPUT.PUT_LINE(v_name);
    END;
    /

    ------------------------------------------------------------
    CREATE OR REPLACE PROCEDURE sp OneINOUTParameter
    (genericParam IN OUT VARCHAR2)
    IS
    BEGIN
      genericParam := 'Hello World INOUT parameter ' || genericParam;
    END;
    /

    SET SERVEROUTPUT ON
    DECLARE
      genericParam VARCHAR2(100) := 'Jimin';
    BEGIN
      sp_OneINOUTParameter(genericParam);
      DBMS_OUTPUT.PUT_LINE(genericParam);
    END;
    /
```

```
SQL> @demo.sql
Hello World INOUT parameter Jimin

PL/SQL procedure successfully completed.

OR

SET SERVEROUTPUT ON
DECLARE
  genericParam VARCHAR2(100) := 'Jimin';

PROCEDURE sp OneINOUTParameter
(genericParam IN OUT VARCHAR2)
IS
BEGIN
  genericParam := 'Hello World INOUT parameter ' || genericParam;
END;

BEGIN
  sp OneINOUTParameter(genericParam);
  DBMS_OUTPUT.PUT_LINE(genericParam);
END;
/

SQL> @demo.sql
Hello World INOUT parameter Jimin

PL/SQL procedure successfully completed.
-------------------------------------------------------------
DECLARE
    a  NUMBER;
PROCEDURE squareNum(x IN OUT NUMBER) IS
BEGIN
    x := x * x;
END;
BEGIN
    a := 23;
    squareNum(a);
    DBMS OUTPUT.PUT LINE(' Square of (23) : ' || a);
END;
/
-------------------------------------------------------------
--13. 복합 파라미터
    CREATE OR REPLACE PROCEDURE test_proc
    (
        p name  IN VARCHAR2,
        p_msg   OUT  VARCHAR2
    )
    IS
    BEGIN
        p_msg := 'My name is ' || p_name;
    END;
    /

    DECLARE
        v_msg   VARCHAR2(100);
    BEGIN
        test_proc('조용필', v_msg);
        DBMS_OUTPUT.PUT_LINE(v_msg);
    END;
    /


--14. 파라미터에 기본값 사용하기
    CREATE OR REPLACE PROCEDURE test_proc
    (
        p_name  VARCHAR2  DEFAULT '송대관'
    )
    IS
    BEGIN
        DBMS OUTPUT.PUT LINE('My name is ' || p name);
    END;
    /

    EXEC test proc();

--15. 저장 프로시저의 매개변수 실행 순서
    CREATE OR REPLACE PROCEDURE test_proc
    (
        p_dan  NUMBER,
    p_max_num  NUMBER
    )
    IS
    i  NUMBER;
    BEGIN
          FOR i IN  1..p max num
    LOOP
     DBMS_OUTPUT.PUT_LINE(p_dan || ' * ' || i || ' = ' || p_dan * i);
    END LOOP;
    END;
    /
    EXEC test_proc(3, 9);  --3단

    CREATE OR REPLACE PROCEDURE test proc
    (
```

```
        p dan  NUMBER  DEFAULT  2,
        p max num  NUMBER   DEFAULT   9
    )
    IS
        i  NUMBER;
    BEGIN
        FOR i IN  1..p_max_num
        LOOP
            DBMS OUTPUT.PUT LINE(p dan || ' * ' || i || ' = ' || p dan * i);
        END LOOP;
    END;
    /
    EXEC test proc(4);   --4단, 즉 값을 한 개만 넣으면 제일 앞의 매개변수로 인식

    --만일 2번째 매개변수로 인식하려면
    EXEC test proc(p max num => 5);


--16. 저장 프로시저 삭제
  DROP PROCEDURE [schema].procedure name;


REM Stored FUNCTION
 --1)오라클에서 기본적으로 제공하는 함수 이외에 사용자가 필요에 따라 만든 함수
 --2)프로시저와 성격이 매우 비슷하지만, 반환값이 있느냐 없느냐가 가장 큰 차이
 --3) 실행시 반드시 하나의 값을 RETURN하기 위해 사용
 --4) 함수 선언에서 Data Type이 있는 RETURN 절을 추가하고 PL/SQL 블록에 적어도 한개 이상의 RETURN문을 포함
 --5)Syntax
    CREATE OR REPLACE FUNCTION [SCHEMA.]function_name
    [(argument1  [mode]  datatype[, argument2  [mode] datatype,...])]
    RETURN data type
    {IS | AS}
        변수 선언부
    BEGIN
        statement1;
    statement2;
    RETURN variable_name;
    END ;

  --실행
  EXECUTE  :variable_name := function_name(argument_list);

-----------------------------------------------------------------------
    CREATE FUNCTION chk_sal
    (v_sal NUMBER)
        RETURN NUMBER
    IS
        v_chk NUMBER;
    BEGIN
        v chk := v sal * 0.01;
        RETURN v_chk;
    END;
    /

    -EXECUTE 명령어에 의해 실행(SQL*Plus 환경)
    variable v_sal NUMBER
    EXECUTE :v sal := chk sal(7900);
    PL/SQL procedure successfully completed.
    SQL> print v_sal

     V SAL
    ----------
        79

    -표현식의 일부로서 호출될 수 있다.
    SQL> SELECT empno, sal, chk_sal(sal)
      2  FROM emp
      3  WHERE deptno = 10;

          EMPNO        SAL CHK_SAL(SAL)
    ---------- ---------- ------------
          7782       2450         24.5
          7839       5000           50
          7934       1300           13

    SQL> SELECT SUM(sal), chk sal(SUM(sal))
      2  FROM emp
      3  WHERE deptno = 10;

      SUM(SAL) CHK_SAL(SUM(SAL))
    ---------- -----------------
          8750              87.5

-----------------------------------------------------------------
    CREATE OR REPLACE FUNCTION tax
    (v value IN NUMBER)
    --함수는 IN argument 만 사용한다.
        RETURN number
    IS
    BEGIN
        RETURN v_value * 0.07;
    END;
    /

    SQL> @demo.sql
```

```
    Function created.
    SQL> variable x number
    SQL> execute :x := tax(100)
    PL/SQL procedure successfully completed.

    SQL> print x

            X
    ----------
              7

    SQL> SELECT sal, tax(sal) FROM emp
    2  WHERE empno = 7900;

            SAL   TAX(SAL)
    ---------- ----------
            950       66.5

    SQL> UPDATE emp SET sal = tax(sal)
    2  WHERE empno = 7900;

    1 row updated.
-------------------------------------------------------------

CREATE OR REPLACE FUNCTION test_func(p_name IN VARCHAR2)
    RETURN VARCHAR2
    IS
        v_name   VARCHAR2(100) := '';
    BEGIN
        v_name := 'My Name is ' || p_name;
        RETURN v name;
   END ;

FUNCTION created.

SQL> SELECT test_func('조성모') AS "이름" FROM dual;

이름
-----------------
My Name is 조성모

OR

SQL> VAR v_name VARCHAR2(100);
SQL> EXEC  :v_name := test_func('설운도');

PL/SQL procedure successfully completed.

SQL> PRINT v name;
V NAME
----------------------------------------
My Name is 설운도

  --4)주의 사항
     --a. SELECT 문에서 사용되는 함수에는 DML  을 포함해서는 안된다.
     --b. UPDATE 나 DELETE 문에서 사용되는 함수에는 같은 테이블에 대한 DML  과 SELECT 문이 포함되서는 안된다.

--보너스는 급여의 200%를 지급한다.

CREATE OR REPLACE FUNCTION cal bonus
(  v empno  IN emp.empno%TYPE  )
RETURN NUMBER
IS
    v sal  NUMBER(7,2);
BEGIN
     SELECT sal
     INTO v_sal
     FROM emp
     WHERE empno = v_empno;

     RETURN v sal * 200;
END ;
/

SQL> VAR v rebonus NUMBER;
SQL> EXECUTE  :v rebonus := cal bonus(7788);

PL/SQL procedure successfully completed.

SQL> PRINT v_rebonus;
SQL> SELECT empno, sal, cal_bonus(7369)
           FROM emp WHERE empno = 7369;

--사원명으로 검색하여 해당 사원의 직급을 얻어 오는 함수를 fun_sel_empname라는 이름으로 작성하시오.
CREATE OR REPLACE FUNCTION fun_sel_empname
(  v ename IN emp.ename%TYPE)
RETURN VARCHAR2
IS
    v_job emp.job%TYPE;
BEGIN
   SELECT job INTO v_job
   FROM emp
   WHERE ename = UPPER(v_ename);

   RETURN v_job;
```

```
END ;
/

SQL>VAR v_job VARCHAR2(10);
SQL>EXEC :v job := fun sel empname('scott');
SQL>PRINT v_job;
SQL>SELECT ename, job, fun_sel_empname('scott')
         FROM emp WHERE ename = UPPER('scott');

   --5)삭제
   DROP FUNCTION function_name;

-emp table에서 이름으로 부서 번호를 검색하는 함수를 작성하시오
--CREATE OR REPLACE   FUNCTION   ename_deptno
--(
--  v ename     IN  emp.ename%TYPE
--)
--RETURN    NUMBER
--IS
--  v deptno     emp.deptno%TYPE;
--BEGIN
--  SELECT deptno
--  INTO  v deptno
--  FROM   emp
--  WHERE ename = UPPER(v_ename);
--RETURN v_deptno;
--END;
--VAR g_deptno  NUMBER
--EXECUTE  : g_deptno := ename_deptno('SCOTT')

--emp table에서 이름을 입력받아 부서번호, 부서명, 급여를 검색하는 function을 작성하시오. 단 부서번호를 RETURN에 사용하시오.
--CREATE OR REPLACE FUNCTION emp_disp
--(
--  v ename     IN  emp.ename%TYPE,
--  v_dname     OUT dept.dname%TYPE,
--  v_sal       OUT emp.sal%TYPE
--)
--RETURN    NUMBER
--IS
--  v_deptno        emp.deptno%TYPE;
--  v dname temp    dept.dname%TYPE;
--  v_sal_temp  emp.sal%TYPE;
--BEGIN
--  SELECT sal, deptno
--  INTO    v sal temp, v deptno
--  FROM emp
--  WHERE ename = UPPER(v_ename);
--  SELECT dname
--  INTO    v dname temp
--  FROM  dept
--  WHERE deptno = v_deptno;
--  v dname := v dname temp;
--  v_sal := v_sal_temp;
--END;
--VAR g_deptno  NUMBER
--VAR g dname   VARCHAR(20)
--VAR g_sal   NUMBER
--EXECUTE  :g_deptno := emp_disp('SCOTT', :g_dname, :g_sal)


REM  Procedure vs Function
--PROCEDURE                          FUNCTION
--PL/SQL 문으로서 실행              식의 일부로서 실행
--RETURN datatype이 없슴          RETURN datatype이 필수
--값을 RETURN 할 수 있슴          값을 RETURN 하는 것이 필수
```