```
REM SQL
--1. 장점
  --1) SQL 질의문 하나로 원하는 데이터를 검색 및 조작할 수 있다.
  --2) 사용자가 이해하기 쉬운 단어로 구성
  --3) 복잡한 로직을 간단하게 작성할 수 있다.
  --4) ANSI 에 의해 문법이 표준화되어 있다.
--2. 단점
  --1) 반복처리를 할 수 없다(LOOP)
  --2) 비교처리를 할 수 없다(IF).
  --3) Error 처리를 할 수 없다(EXCEPTION).
  --4) SQL 문을 캡슐화할 수 없다.
  --5) 변수선언을 할 수 없다.
  --6) 실행할 때마다 분석작업 후 실행한다.
  --7) Network Traffic 을 유발한다.
  --8) SQL 문 자체는 비 절차적 언어이므로, 여러 개의 질의문 사이에 연결이나 절차가 있어야 할 때에는 사용할 수 없다.
  --9) 실제 프로그래밍에서는 다른 언어를 사용해서 각각의 SQL 문들을 서로 연관되도록 하고 절차적 또는 순차적인 단계를 가지고 SQL문이 실행되도록 해야 한다.
  --10) 다른 언어를 이용해서 처리해도 되고, 오라클 자체적으로는 PL/SQL 을 사용한다.
  --11) 다른 RDBMS 에서는 사용할 수 없다.

REM PL/SQL
--1. 개요
  --1) Stands for Procedural Language Extension to Structured Query Language
  --2) SQL 문의 제한을 극복하기 위해 Oracle 에서 SQL 언어에 절차적인 프로그래밍 언어를 가미해 생성
  --3) 일반 프로그래밍의 언어적인 요소를 거의 다 가지고 있다.
  --4) SQL문과 결합하여 데이터 트랙잭션 처리나 정보 보호, 데이터에 대한 보안, 예외 처리 기능, 객체지향 등 데이터베이스와 관련된 중요한 모든 기능 지원
  --5) 파스칼의 변형인 Ada에 기반
  --6) Oracle 6 버전에서 시작

--2. 장점
  --1) PL/SQL은 네트워크 트래픽을 감소시킴으로써 시스템의 응답시간을 향상시킨다.
  --2) 여러 SQL문을 처리하는 것보다 PL/SQL은 응용 프로그램 성능을 향상시킨다.
  --3) PL/SQL은 SQL로 얻을 수 없는 절차적 언어의 기능을 가지고 있다.
  --4) SQL 언어와 비슷하므로 접근하기가 쉬어 개발에 대한 시간이 적게 들기 때문에 생산성이 뛰어나다.
  --5) 이식성 --> 이미 만들어진 PL/SQL 프로그램은 다른 오라클 서버에서 변환 작업 없이 사용가능
  --6) 보안

--3. 특징
  --1) 프로그램 개발시 모듈화
    --a. 논리적 문장들을 그룹화
    --b. 복잡한 프로그램 모듈을 그룹화가능
  --2) 변수 선언
  --3) 상수 선언
  --4) 절차적 구조로 된 프로그래밍
    --조건문, 반복문
  --5) ERROR 처리 가능
    --Exception 처리가능
  --6) 성능향상

--4. 구조
  --DECLARE --> 선언부 (선택)
  --BEGIN --> 실행부 시작 (필수)
  --EXCEPTION  --> 에러처리부 (선택)
  --END;   --> 실행부 끝 (필수)

    /*
        1. PL/SQL 문을 SQL*Plus 에서 직접 수행할 때에는
        SET SERVEROUTPUT ON... SET SERVEROUTPUT OFF 를
        넣어야 한다.
            SET SERVEROUTPUT ON
            BEGIN
                DBMS OUTPUT.PUT LINE('Hello, World!');
            END;
            /     <-- 반드시 넣어야 함.

        2. SQL Developer에서는 넣지 않는다.
        BEGIN
            DBMS_OUTPUT.PUT_LINE('Hello, World!');
        END;
        View > Dbms Output 메뉴 Click
        Dbms Output 창이 나타나면 제일 왼쪽의 +버튼(Enable DBMS_OUTPUT for Connection(Ctrl + N)) Click
        Select Connection 창이 나타나면 Connection 선택 후 OK Click
        Run Script(F5) 클릭. 결과는 Dbms Output 창으로 출력한다.
        출력전에 Clear(Ctrl + D) 를 클릭하여 Dbms Output 창을 클리어시킨다.
        실행시 'Script Output' 창이 나타나면 제목위의 아래화살표를 클릭하여 숨긴다.

        3. ed file name.sql 은 프롬프트의 현재위치에서 file name.sql 을 생성한다.
        이때 sql 확장자는 넣지 않아도 된다. 저장 후 닫고 SQL*Plus 에서 @ or START file_name[.sql]
        실행하면 된다.
        SET SERVEROUTPUT ON
        START | @ file name[.sql]
            BEGIN
                DBMS_OUTPUT.PUT_LINE('Hello, World!');
            END;
            /     <----반드시 넣어야 함.
        PL/SQL procedure successfully completed.

        4. SQL Explorer 에서는 SQL Editor 에서
        BEGIN
            DBMS_OUTPUT.PUT_LINE('Hello, World!');
        END;
        입력후, 3번째 런 버튼(Execute current SQL and display results in batch mode.)를 클릭
        그러면, 아래 Message 뷰에 결과 출력
```

```
    */

    [SET SERVEROUTPUT ON]
    BEGIN
        DBMS OUTPUT.PUT LINE('Hello, World!');
    END;
    [SET SERVEROUTOUT OFF]

    [SET SERVEROUTPUT ON]
    DECLARE
        counter  NUMBER := 1;
    BEGIN
        IF counter IS NULL THEN
            DBMS_OUTPUT.PUT_LINE('Result : Counter is Null');
        ELSE
            counter := counter + 1;
        END IF;
            DBMS_OUTPUT.PUT_LINE('Result = ' || counter);
    END;
    /

    SET SERVEROUTPUT ON
    DECLARE
        counter   NUMBER;
        i         NUMBER;
    BEGIN
        FOR  i  IN  1..10  LOOP
            counter := (2 * i);
            DBMS_OUTPUT.PUT_LINE(' 2 x ' || i || ' = ' || counter);
        END LOOP;
    END;
    /

--5. GuideLine
   --1) PL/SQL Block 에서는 한 문장이 종료될 때마다 세미클론(;)을 기술한다.
   --2) 실행부 마지막 (END) 뒤에도 세미클론(;)을 사용하여 하나의 블록이 끝났다는 것을 기술해야 한다.
   --3) 바로 SQL*Plus 에서 실행할 수 있고, 편집기를 통해서 편집 및 실행할 수도 있다.
   --4) SQLBuffer 로 작업하면 마지막에 '/'을 넣어서 실행하고, 파일로 저장 및 편집해서 실행하면 start, @
   경로 를 이용한다.
   --5) DDL은 사용할 수 없다.
   --6) GRANT, REVOKE 도 사용할 수 없다.
   --7) 직접 데이터를 검색하여 사용할 수 없다.
     --즉 SELECT로 추출되어야 하는 ROW의 수는 1건이어야 한다.
     --따라서 데이터가 1건으로 검색될 수 있는 조건은 아래와 같다.
       --a. PRIMARY KEY 또는 UNIQUE로 검색조건 사용
       --b. 단일 그룹 함수
       --c. ROWNUM = 1을 검색조건으로 사용
   --8) 식별자는 최대 30문자로 작성할 수 있다.
   --9) 식별자는 테이블 또는 칼럼 이름과 동일할 수 없다.
   --10) 식별자는 알파벳으로 시작해야 한다.
   --11) 문자와 날짜 타입은 단일 인용부호(홑따옴표)로 표시한다.
   --12) 주석은 싱글주석 (--), 멀티라인주석(/* ~ */) 로 표시한다.
   --13) 블록 내에서는 CREATE, LEAST, DECODE, 그룹함수를 사용할 수 없다.

--6. PL/SQL 블록의 유형
   --1) Anonymous 형
     /*
    [DECLARE]
    BEGIN
    [EXCEPTION]
    END;
     */

   --2) Stored Procedure 형
     /*
    CREATE OR REPLACE produre_name
    IS
    BEGIN
    [EXCEPTION]
    END;
     */

   --3) Stored Function 형
     /*
    CREATE function function name
    RETURN datatype
    IS
    BEGIN
        return value;
    [EXCEPTION]
    END;
     */

--7. 종류
   --1) 익명 블록(Anonymous)
   --2) Stored(저장, 내장) 프로시저, 함수
   --3) Package
   --4) Trigger
   --5) Cursor

--8. 변수(상수)선언
   --1) 스칼라 변수
   --2) 참조 변수
   --3) Syntax
      --식별자  [CONSTANT] data_type [NOT NULL] [ := DEFALUT expression]
```

```
ACCEPT p_empno PROMPT ' Employee Number : '
ACCEPT p_name PROMPT ' Name : '
ACCEPT p_sal PROMPT ' Salary : '
ACCEPT p_deptno PROMPT ' Dept No. : '
DECLARE
    v_empno     NUMBER(4) := &p_empno;
    v_name      VARCHAR2(10) := UPPER('&p_name');
    v_sal        NUMBER(7,2) := &p_sal;
    v_deptno    NUMBER(2) := &p_deptno;
BEGIN
    IF v_deptno = 10 THEN
            v_sal := v_sal * 1.13;
    ELSIF   v_deptno = 20 THEN
        v_sal := v_sal * 1.15;
    ELSIF   v_deptno = 30 THEN
            v_sal := v_sal * 1.18;
    END IF;
    INSERT INTO emp(empno, ename, sal, deptno)
    VALUES(v_empno, v_name, v_sal, v_deptno);
    COMMIT;
END;
/

--4) 예
DECLARE
    v_hiredate      DATE;
    v_deptno            NUMBER(2) NOT NULL := 10;
    v_loc               VARCHAR2(13) := 'PUSAN';
    c_comm          CONSTANT  NUMBER := 1400;

--5) GuideLine
  --a. 가능하면 현재 사용하고 있는 테이블의 컬럼이름을 이용한다.
  --b. 명명규칙을 가급적이면 지키자(v_xxx)
  --c. NOT NULL 이면 값을 초기화한다.
  --d. 대입연산자( := )을 사용하거나 DEFAULT 을 사용하여 값을 초기화할 수 있다.
  --e. 한 라인에 한개의 변수를 선언한다.
  --f. 상수를 선언할 때에는 타입보다 CONSTANT 를 먼저 기술하고, 반드시 초기화 해야 한다.

--6) Scalar 변수
  --a. 하나의 SCLAR 변수에 사용자의 임시 데이터를 하나만 저장할 수 있는 타입
  --b. BOOLEAN : TRUE, FALSE, NULL
  --c. BINARY_INTEGER(PLS_INTEGER, 10g 부터) : -2147483648 ~ 2147483647
  --d. NUMBER : 정수나 소수점을 포함하는 숫자 데이터를 저장할 때
  --e. CHAR : 고정길이 문자
  --f. VARCHAR2 : 가변길이 문자
  --g. LONG(CLOB) : 2GB(4GB)까지의 대용량 고정길이 문자
  --h. LONG RAW (BLOB): 2GB(4GB) dml Binary 데이터를 저장할 때
  --i. DATE(TIMESTAMP) : 날짜와 시간
  --j. 예
DECLARE
     v_job              VARCHAR2(9);
     v_count           BINARY_INTEGER := 0;
     v_total_sal        NUMBER(9, 1) := 0;
     v_order_date     DATE := SYSDATE + 7;
     c_tax              CONSTANT NUMBER(3,2) := 8.25;
     v_valid            BOOLEAN NOT NULL := TRUE;
     v_gender         CHAR(1) ;
     c_female     CONSTANT CHAR(1) := 'F';
     c_male        CONSTANT CHAR(1) DEFAULT 'M';
     c_limit         CONSTANT PLS_INTEGER := 10;
     c_current_date  CONSTANT DATE := SYSDATE;


SET SERVEROUTPUT ON
DECLARE
    V_EMPNO  NUMBER(4);
    V_ENAME  VARCHAR2(10);
BEGIN
    V_EMPNO := 7788;
    V_ENAME := UPPER('scott');
    DBMS_OUTPUT.PUT_LINE('사번 | 이름');
    DBMS_OUTPUT.PUT_LINE('----------------------');
    DBMS_OUTPUT.PUT_LINE(V_EMPNO || ' | ' || V_ENAME);
END;
/

--7) %TYPE 변수
  --a. 예
DECLARE
    v_empno     emp.empno%TYPE
    v_ename      emp.ename%TYPE
    v_deptno     dept.deptno%TYPE

SET SERVEROUTPUT ON
DECLARE
    V_EMPNO  EMP.EMPNO%TYPE;
    V_ENAME  EMP.ENAME%TYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('사번 | 이름');
    DBMS_OUTPUT.PUT_LINE('----------------------');
    SELECT empno, ename
    INTO V_EMPNO, V_ENAME
    FROM emp
    WHERE ENAME = UPPER('scott');
```

```
        DBMS_OUTPUT.PUT_LINE(V_EMPNO || ' | ' || V_ENAME);
    END;
    /

--사번을 입력받은 후, 사원의 이름과 사원의 입사날짜를 출력하는 PL/SQL문을 작성하시오.
    ACCEPT p_empno   PROMPT  ' Employee Number : '
    DECLARE
        v_empno    emp.empno%TYPE := &p_empno;
        v_ename    emp.ename%TYPE;
        v_hiredate  emp.hiredate%TYPE;
    BEGIN
        SELECT ename, hiredate
        INTO  v_ename, v_hiredate
        FROM emp
        WHERE empno = v_empno;

        DBMS_OUTPUT.PUT_LINE('Employee No.      Name     Hiredate');
        DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
        DBMS_OUTPUT.PUT_LINE(v_empno || '      ' || v_ename || '    ' || v_hiredate);
    END;
    /

   --8)%ROWTYPE
     --a. 테이블 또는 뷰의 여러가지 열들을 RECORD로 사용하기 위해
     --b. 테이블 이름 뒤에 %ROWTYPE을 붙여 선언한다.
     --c. 장점
       -- 알지 못하는 테이블의 칼럼의 형식이 자동으로 부여된다.
       -- 실행시 컬럼의 갯수와 데이터 형식을 몰라도 된다.
       -- 칼럼의 수가 많을 때 효과적

-------사원 테이블에서 이름을 입력받아 각 사원의 정보를 출력하는 SCRIPT를 작성하시오.

    SET SERVEROUTPUT ON
    ACCEPT p_ename  PROMPT 'Enter a Name : '
    DECLARE
        v_ename    emp.ename%TYPE := '&p_ename';
        emp_record    emp%ROWTYPE;
    BEGIN
        SELECT *
        INTO      emp_record
        FROM emp
        WHERE  ename = UPPER(v_ename);

        DBMS_OUTPUT.PUT_LINE('Employee Number : ' || TO_CHAR(emp_record.empno));
        DBMS_OUTPUT.PUT_LINE('Name : ' || emp_record.ename);
        DBMS_OUTPUT.PUT_LINE('Job : ' || emp_record.job);
        DBMS_OUTPUT.PUT_LINE('Hiredate : ' || TO_CHAR(emp_record.hiredate, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Salary : ' || TO_CHAR(emp_record.sal, '$999,999.00'));
    END;
    /
    SET SERVEROUTPUT OFF

   --9). 테이블 TYPE
     --1) 여러 ROW  처리하기 위한 방법
     --2) Syntax
    TYPE table_type_name IS TABLE OF {column_type | varaible%TYPE | table_name.column%TYPE} [NOT NULL]
    [INDEX BY BINARY_INTEGER];
    identifier table_type_name;

    --배열 형태의 테이블 타입 선언
    TYPE ENAME_TABLE_TYPE IS TABLE OF EMP.ENAME%TYPE
    INDEX BY BINARY_INTEGER;
    --테이블 변수 선언
    ENAME_TABLE  ENAME_TABLE_TYPE;
    --인덱스로 사용할 변수 선언, 인덱스는 양의 정수 값 저장
    I BINARY_INTEGER := 0;
    --반복문을 통해 SELECT 문으로 얻어진 칼럼 값들을 테이블 변수에 저장
    FOR K IN (SELECT ENAME FROM EMP) LOOP
        I := I + 1;
    ENAME_TABLE(I) := K.ENAME;
    END LOOP;
    --출력
    FOR J IN 1..I LOOP
        DBMS_OUTPUT.PUT_LINE(ENAME_TABLE(J));
    END LOOP;

-----TABLE  변수를 이용하여 EMP 테이블에서 이름과 업무를 출력하라.
    SET SERVEROUTPUT ON
    DECLARE
        TYPE ENAME_TABLE_TYPE IS TABLE OF EMP.ENAME%TYPE
        INDEX BY BINARY_INTEGER;
        TYPE JOB_TABLE_TYPE IS TABLE OF EMP.JOB%TYPE
        INDEX BY BINARY_INTEGER;

        ENAME_TABLE ENAME_TABLE_TYPE;
        JOB_TABLE JOB_TABLE_TYPE;

        I BINARY_INTEGER := 0;
    BEGIN
        FOR K IN (SELECT ENAME, JOB FROM EMP) LOOP
            I := I + 1;
            ENAME_TABLE(I) := K.ENAME;
            JOB_TABLE(I) := K.JOB;
        END LOOP;
```

```
        FOR J IN 1..I LOOP
            DBMS_OUTPUT.PUT_LINE(RPAD(ENAME_TABLE(J), 12) || ' | ' || RPAD(JOB_TABLE(J), 9));
        END LOOP;
    END;
    /
   --10) PL/SQL
--9. 조건문
   --1) IF 문
      -- Syntax
      /*
    IF 조건 THEN
          처리문;
    ENF IF;
      */
      /*
    IF 조건 THEN
          처리문1;
    ELSE
           처리문2;
    END IF;
      */
      /*
    IF 조건1 THEN
        처리문1;
    ELSIF  조건2  THEN
          처리문2;
    ELSIF  조건3  THEN
        처리문3;
    ...
    ELSE
        처리문N;
    END IF;
      */

------성적관리프로그램
    ACCEPT t_hakbun PROMPT '학번 : ';
    ACCEPT t_name PROMPT '이름 : ';
    ACCEPT t_kor PROMPT '국어 : ';
    ACCEPT t_eng PROMPT '영어 : ';
    ACCEPT t_mat PROMPT '수학 : ';
    ACCEPT t_edp PROMPT '전산 : ';
    DECLARE
        v_hakbun  CHAR(4) := '&t_hakbun';
        v_name  VARCHAR2(20)  := '&t_name';
        v_kor NUMBER(3)  := &t_kor;
        v_eng NUMBER(3)  := &t_eng;
        v_mat NUMBER(3)  := &t_mat;
        v_edp NUMBER(3)  := &t_edp;
        v_tot NUMBER(3);
        v_avg NUMBER(5,2);
        v_grade CHAR(1);
    BEGIN
        v_tot := v_kor + v_eng + v_mat + v_edp;
        v_avg := v_tot / 4.;
        IF v_avg >= 90 AND v_avg <= 100 THEN
            v_grade := 'A';
        ELSIF v_avg > 80 AND v_avg < 90 THEN
            v_grade := 'B';
        ELSIF v_avg > 70 AND v_avg < 80 THEN
            v_grade := 'C';
        ELSIF v_avg > 60 AND v_avg < 70 THEN
            v_grade := 'D';
        ELSE
            v_grade := 'F';
        END IF;
        DBMS_OUTPUT.PUT_LINE('   *********성적관리프로그램**********');
        DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('학번 | 이름 | 국어 | 영어 | 수학 | 전산 | 총점 | 평균 | 평점');
        DBMS_OUTPUT.PUT_LINE('-------------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE(v_hakbun || ' | ' || v_name || ' | ' || v_kor || ' | ' ||
        v_eng || ' | ' || v_mat || ' | ' || v_edp || ' | ' || v_tot || ' | ' ||
        v_avg || ' | ' || v_grade);
    END;
    /

--이름, 급여, 부서번호를 입력받아 사원테이블에 자료를 등록하는 PL/SQL 문을 완성하시오.
--단, 부서번호가 10번이면 입력한 급여의 20%를 추가하고,
--초기값이 9000부터 9999까지 1씩 증가하는 SEQUENCE를 작성하여 사용하시오.

    CREATE SEQUENCE emp_seq
        START WITH 9000
        INCREMENT BY 1
        MAXVALUE   9999;

    ACCEPT p_ename    PROMPT  ' Name : '
    ACCEPT p_sal  PROMPT  ' Salary  : '
    ACCEPT p_deptno PROMPT 'Department Number : '
    DECLARE
        v_ename    emp.ename%TYPE := UPPER('&p_ename');
        v_sal         emp.sal%TYPE := &p_sal;
        v_deptno   emp.deptno%TYPE := &p_deptno;
    BEGIN
        IF  v_deptno = 10 THEN
            v_sal := v_sal * 1.2;
        END IF;
```

```
472
473         INSERT INTO emp(empno, ename, sal, deptno)
474         VALUES(emp_seq.NEXTVAL, v_ename, v_sal, v_deptno);
475         COMMIT;
476     END;
477     /
478
479 --이름을 입력받아서 그 사람의 업무가 MANAGER, ANALYST 이면 급여가 50% 가산하여 갱신하고, 업무가 MANAGER,
    ANALYST 가 아니면 20% 가산하는 PL/SQL 문을 작성하시오.
480
481     ACCEPT p_name  PROMPT  'Name : '
482     DECLARE
483         v ename     emp.ename%TYPE := UPPER('&p name');
484         v_job          emp.job%TYPE;
485         v_sal          emp.sal%TYPE;
486     BEGIN
487         SELECT job, sal
488         INTO v_job, v_sal
489         FROM emp
490         WHERE ename = v ename;
491
492         IF v_job IN ('MANAGER', 'ANALYST') THEN
493             v sal := v sal * 1.5;
494         ELSE
495             v_sal := v_sal * 1.2;
496         END IF;
497
498         UPDATE emp
499         SET sal = v_sal
500         WHERE ename = v_ename;
501         COMMIT;
502         DBMS_OUTPUT.PUT_LINE('Update Success.');
503     END;
504     /
505    --2) CASE문
506     DECLARE
507         grade    CHAR(1);
508     BEGIN
509         grade := 'B';
510
511         CASE grade
512             WHEN 'A' THEN
513                 DBMS_OUTPUT.PUT_LINE('Excellent');
514             WHEN 'B' THEN
515                 DBMS_OUTPUT.PUT_LINE('Good');
516             WHEN 'C' THEN
517                 DBMS_OUTPUT.PUT_LINE('Fair');
518             WHEN 'D' THEN
519                 DBMS OUTPUT.PUT LINE('Poor');
520             ELSE
521                 DBMS_OUTPUT.PUT_LINE('Not Found');
522         END CASE;
523     END;
524     /
525
526 --10. 반복문
527    --1) LOOP ~ END LOOP --> do ~ while;
528     DECLARE
529         i    NUMBER := 1;
530     BEGIN
531         LOOP
532             DBMS_OUTPUT.PUT(i || '    ');
533             i := i + 1;
534             IF i > 5 THEN
535                 EXIT;
536             END IF;
537         END LOOP;
538         DBMS OUTOUT.NEW LINE();
539     END;
540     /
541
542 ---BASIC LOOP를 이용하여 1부터 5까지 출력하기
543     SET SERVEROUTPUT ON
544     DECLARE
545          N NUMBER := 1;
546     BEGIN
547          LOOP
548           DBMS_OUTPUT.PUT(N || '    ');
549           N := N + 1;
550           IF N > 5 THEN
551                EXIT;
552           END IF;
553          END LOOP;
554          DBMS_OUTOUT.NEW_LINE();
555     END;
556     /
557
558 --아래와 같은 출력결과를 얻을 수 있도록 LOOP ~ END LOOP 로 완성하시오.
559     --2
560     --4
561     --6
562     --8
563     --10
564     DECLARE
565         v_result     INTEGER;
```

```
        i             INTEGER :=1;
    BEGIN
        LOOP
            v_result := 2 * i;
            IF i > 5 THEN
                EXIT;
            ELSE
                DBMS_OUTPUT.PUT_LINE(v_result);
            END IF;
            i := i + 1;
        END LOOP;
    END;
    /

--구구단
    DECLARE
        j   NUMBER;
        i       NUMBER;
    BEGIN
        i := 1;
        LOOP
            IF (i > 9) THEN
                EXIT;
            END IF;
            j := 2;
            LOOP
                IF (j > 9) THEN
                    EXIT;
                END IF;
                DBMS_OUTPUT.PUT(j || ' * ' || i || ' = ' || j * i || '   ');
                j := j + 1;
            END LOOP;
            i := i + 1;
            DBMS OUTPUT.NEW LINE();
        END LOOP;
        DBMS_OUTPUT.NEW_LINE();
    END;
    /

   --2) FOR LOOP --> for()
    SET SERVEROUTPUT ON
    DECLARE
        i    NUMBER;
    BEGIN
        FOR i  IN 1..5 LOOP
            DBMS OUTPUT.PUT LINE(i);
        END LOOP;
    END;
    /

    DECLARE
        j   NUMBER;
        i   NUMBER;
    BEGIN
        FOR i IN 1 .. 9 LOOP
            FOR j IN 2 .. 9 LOOP
                DBMS_OUTPUT.PUT(j || ' * ' || i || ' = ' || j * i || '   ');
            END LOOP;
            DBMS_OUTPUT.NEW_LINE();
        END LOOP;
    END;
    /

--FOR LOOP 를 이용하여 부서테이블을 출력하시오.
    SET SERVEROUTPUT ON
    DECLARE
          V_DEPT  dept%ROWTYPE
    BEGIN
          DBMS_OUTPUT.PUT_LINE('부서 번호 | 부서명 | 지역명');
          DBMS_OUTPUT.PUT_LINE('-------------------------------------------------');
          FOR CNT IN 1..4 LOOP
           SELECT *
           INTO V_DEPT
           FROM dept
           WHERE deptno = 10 * CNT;

           DBMS_OUTPUT.PUT_LINE(V_DEPT.deptno || ' | ' || V_DEPT.dname || ' | ' || V_DEPT.loc);
          END LOOP;
    END;
    /

   --3)WHILE LOOP --> while()
    DECLARE
        V_NUMBER     NUMBER :=2;
        V_CNT             NUMBER :=1;
    BEGIN
        WHILE V CNT < 10
        LOOP
            DBMS_OUTPUT.PUT_LINE(TO_CHAR(V_NUMBER) || ' * ' || TO_CHAR(V_CNT) || ' = ' || TO_CHAR(V_NUMBER *
            V CNT));
            V_CNT := V_CNT + 1;
        END LOOP;
    END;
    /
```