■STEP1. BASIC

Q001

- 다음과 같이 프로시저를 생성하시오.

```
Procedure PRO_NOPARAM이(가) 컴파일되었습니다.
```

```
CREATE OR REPLACE PROCEDURE pro_noparam
IS
   V_EMPNO NUMBER(4) := 7788;
   V_ENAME VARCHAR2(10);
BEGIN
   V_ENAME := 'SCOTT';
   DBMS_OUTPUT.PUT_LINE('V_EMPNO : ' || V_EMPNO);
   DBMS_OUTPUT.PUT_LINE('V_ENAME : ' || V_ENAME);
END;
/
```

Q002

- 위에서 생성한 프로시저를 실행하시오.

```
V_EMPNO : 7788
V_ENAME : SCOTT

PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
SET SERVEROUTPUT ON;

EXECUTE pro_noparam;
```

Q003

- 익명블록에서 프로시저를 실행하시오.

```
V_EMPNO : 7788
V_ENAME : SCOTT


PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
BEGIN
   pro_noparam;
END;
/
```

Q004

- USER-SOURCE를 통해 프로시저를 확인하시오.

	NAME	TYPE	LINE	TEXT
1	PRO_NOPARAM	PROCEDURE	1	PROCEDURE pro_noparam
2	PRO_NOPARAM	PROCEDURE	2	IS
3	PRO_NOPARAM	PROCEDURE	3	V_EMPNO NUMBER(4) := 7788;
4	PRO_NOPARAM	PROCEDURE	4	V_ENAME VARCHAR2(10);
5	PRO_NOPARAM	PROCEDURE	5	BEGIN
6	PRO_NOPARAM	PROCEDURE	6	V_ENAME := 'SCOTT';
7	PRO_NOPARAM	PROCEDURE	7	DBMS_OUTPUT.PUT_LINE('V_EMPNO : ' || V_EMPNO);
8	PRO_NOPARAM	PROCEDURE	8	DBMS_OUTPUT.PUT_LINE('V_ENAME : ' || V_ENAME);
9	PRO_NOPARAM	PROCEDURE	9	END;

```sql
SELECT *
  FROM USER_SOURCE
 WHERE NAME = 'PRO_NOPARAM';
```

Q005

- USER-SOURCE를 통해 프로시저를 확인하시오.

	TEXT
1	PROCEDURE pro_noparam
2	IS
3	V_EMPNO NUMBER(4) := 7788;
4	V_ENAME VARCHAR2(10);
5	BEGIN
6	V_ENAME := 'SCOTT';
7	DBMS_OUTPUT.PUT_LINE('V_EMPNO : ' || V_EMPNO);
8	DBMS_OUTPUT.PUT_LINE('V_ENAME : ' || V_ENAME);
9	END;

```sql
SELECT TEXT
  FROM USER_SOURCE
 WHERE NAME = 'PRO_NOPARAM';
```

Q006

- 프로시저를 삭제하시오.

  Procedure PRO_NOPARAM이(가) 삭제되었습니다.

```sql
DROP PROCEDURE PRO_NOPARAM;
```

Q007

- 프로시저에 파라미터를 지정하시오.

  Procedure PRO_PARAM_IN이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PROCEDURE pro_param_in
(
   param1 IN NUMBER,
   param2 NUMBER,
   param3 NUMBER := 3,
   param4 NUMBER DEFAULT 4
)
IS

BEGIN
   DBMS_OUTPUT.PUT_LINE('param1 : ' || param1);
   DBMS_OUTPUT.PUT_LINE('param2 : ' || param2);
   DBMS_OUTPUT.PUT_LINE('param3 : ' || param3);
   DBMS_OUTPUT.PUT_LINE('param4 : ' || param4);
END;
/
```

Q008

- 파라미터를 입력하여 프로시저를 사용하시오.

  param1 : 1
  param2 : 2
  param3 : 9
  param4 : 8

  PL/SQL 프로시저가 성공적으로 완료되었습니다.

```
EXECUTE pro_param_in(1,2,9,8);
```

Q009

- 기본값이 지정된 파라미터 입력을 제외하고 프로시저를 사용하시오.
  param1 : 1
  param2 : 2
  param3 : 3
  param4 : 4

  PL/SQL 프로시저가 성공적으로 완료되었습니다.

```
EXECUTE pro_param_in(1, 2);
```

Q010

- 실행에 필요한 개수보다 적은 파라미터를 입력하여 프로시저를 실행하시오. (에러발생)
  PLS-00306: wrong number or types of arguments in call to 'PRO_PARAM_IN'
  ORA-06550: line 1, column 7:
  PL/SQL: Statement ignored

  https://docs.oracle.com/error-help/db/ora-06550/

  More Details :
  https://docs.oracle.com/error-help/db/ora-06550/
  https://docs.oracle.com/error-help/db/pls-00306/

```
EXECUTE pro_param_in(1);
```

Q011

- 파라미ㅌ이름을 활용하여 프로시저값을 입력하시ㅗㅇ.
  param1 : 10
  param2 : 20
  param3 : 3
  param4 : 4

  PL/SQL 프로시저가 성공적으로 완료되었습니다.

```
EXECUTE pro_param_in(param1 => 10, param2 => 20);
```

Q012

- OUT 모드파라미터를 정의하시오.

  Procedure PRO_PARAM_OUT이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PROCEDURE pro_param_out
(
   in_empno IN EMP.EMPNO%TYPE,
   out_ename OUT EMP.ENAME%TYPE,
   out_sal OUT EMP.SAL%TYPE
)
IS

BEGIN
   SELECT ENAME, SAL INTO out_ename, out_sal
     FROM EMP
    WHERE EMPNO = in_empno;
END pro_param_out;
/
```

Q013

- OUT 모드 파라미터를 사용하시오.

```
ENAME : SCOTT
SAL : 3000


PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
DECLARE
   v_ename EMP.ENAME%TYPE;
   v_sal EMP.SAL%TYPE;
BEGIN
   pro_param_out(7788, v_ename, v_sal);
   DBMS_OUTPUT.PUT_LINE('ENAME : ' || v_ename);
   DBMS_OUTPUT.PUT_LINE('SAL : ' || v_sal);
END;
/
```

Q014

- IN OUT 모드 파라미터를 정의 하시오.

  Procedure PRO_PARAM_INOUT이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PROCEDURE pro_param_inout
(
   inout_no IN OUT NUMBER
)
IS

BEGIN
   inout_no := inout_no * 2;
END pro_param_inout;
/
```

Q015

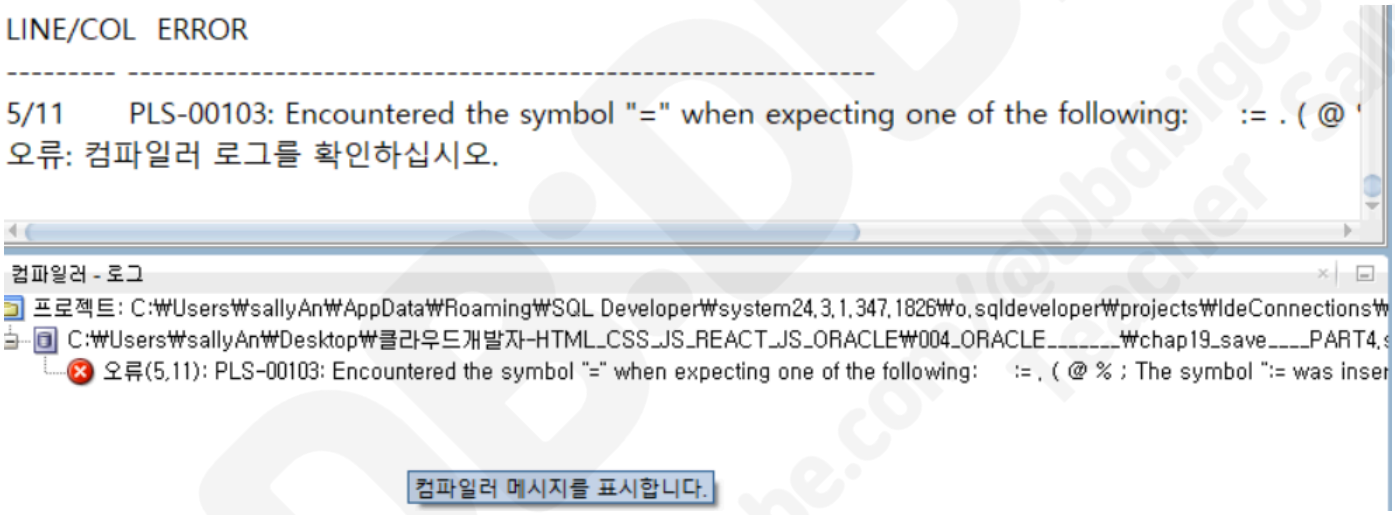
- IN OUT 모드 파라미터를 사용하시오.
  no : 10

  PL/SQL 프로시저가 성공적으로 완료되었습니다.

```
DECLARE
   no NUMBER;
BEGIN
   no := 5;
   pro_param_inout(no);
   DBMS_OUTPUT.PUT_LINE('no : ' || no);
END;
/
```

Q016

- 생성할때 오류가 발생하는 프로시저를 작성하시오.




```
CREATE OR REPLACE PROCEDURE pro_err
IS
   err_no NUMBER;
BEGIN
   err_no = 100;
   DBMS_OUTPUT.PUT_LINE('err_no : ' || err_no);
END pro_err;
/
```

Q017

- SHOW ERRORS 명령어로 오류를 확인하시오.

```
PROCEDURE SCOTT.PRO_ERR에 대한 오류:

LINE/COL ERROR
-------- -----------------------------------------------------------------
5/11     PLS-00103: Encountered the symbol "=" when expecting one of the following:
         := . ( @ % ;
         The symbol ":= was inserted before "=" to continue.
```

```
SHOW ERRORS;
```

Q018

- USER_ERRORS 로 오류를 확인하시오.

	NAME	TYPE	SEQUENCE	LINE	POSITION	TEXT
1	PRO_ERR	PROCEDURE	1	5	11	PLS-00103: Encountered the symbol "=" whe

```
SELECT *
  FROM USER_ERRORS
 WHERE NAME = 'PRO_ERR';
```

Q019

- 다음과 같이 함수를 생성하시오.

  Function FUNC_AFTERTAX이(가) 컴파일되었습니다.

```
CREATE OR REPLACE FUNCTION func_aftertax(
   sal IN NUMBER
)
RETURN NUMBER
IS
   tax NUMBER := 0.05;
BEGIN
   RETURN (ROUND(sal - (sal * tax)));
END func_aftertax;
/
```

Q020

- PL/SQL에서 함수를 사용하시오.

  after-tax income : 2850

  PL/SQL 프로시저가 성공적으로 완료되었습니다.

```
DECLARE
   aftertax NUMBER;
BEGIN
   aftertax := func_aftertax(3000);
   DBMS_OUTPUT.PUT_LINE('after-tax income : ' || aftertax);
END;
/
```

Q021

- sql에서 함수를 사용하시오 .

	FUNC_AFTERTAX(3000)
1	2850

```
SELECT func_aftertax(3000)
  FROM DUAL;
```

Q022

- 함수에 테이블 데이터를 사용하시오.

	EMPNO	ENAME	SAL	AFTERTAX
1	7839	KING	5000	4750
2	7698	BLAKE	2850	2708
3	7782	CLARK	2450	2328
4	7566	JONES	2975	2826
5	7654	MARTIN	1250	1188
6	7499	ALLEN	1600	1520
7	7844	TURNER	1500	1425
8	7900	JAMES	950	903
9	7521	WARD	1250	1188
10	7902	FORD	3000	2850
11	7369	SMITH	800	760
12	7788	SCOTT	3000	2850
13	7876	ADAMS	1100	1045
14	7934	MILLER	1300	1235

```
SELECT EMPNO, ENAME, SAL, func_aftertax(SAL) AS AFTERTAX
  FROM EMP;
```

Q023

- 함수를 삭제하시오.

  Function FUNC_AFTERTAX이(가) 삭제되었습니다.

```
DROP FUNCTION func_aftertax;
```

Q024

- 다음과 같이 패키지를 생성하시오.

  Package PKG_EXAMPLE이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PACKAGE pkg_example
IS
   spec_no NUMBER := 10;
   FUNCTION func_aftertax(sal NUMBER) RETURN NUMBER;
   PROCEDURE pro_emp(in_empno IN EMP.EMPNO%TYPE);
   PROCEDURE pro_dept(in_deptno IN DEPT.DEPTNO%TYPE);
END;
/
```

Q025

- 다음과 같이 패키지 명세를 확인하시오.

	TEXT
1	PACKAGE pkg_example
2	IS
3	spec_no NUMBER := 10;
4	FUNCTION func_aftertax(sal NUMBER) RETURN NUMBER;
5	PROCEDURE pro_emp(in_empno IN EMP.EMPNO%TYPE);
6	PROCEDURE pro_dept(in_deptno IN DEPT.DEPTNO%TYPE);
7	END;

```
SELECT TEXT
  FROM USER_SOURCE
 WHERE TYPE = 'PACKAGE'
   AND NAME = 'PKG_EXAMPLE';
```

Q026

- DESC를 이용하여 패키지 명세를 확인하시오.

```
PROCEDURE PRO_EMP
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 IN_EMPNO                       NUMBER                  IN
```

```
DESC pkg_example;
```

Q027

- 다음과 같이 패키지 본몬을 생성하시오.

  Package Body PKG_EXAMPLE이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PACKAGE BODY pkg_example
IS
   body_no NUMBER := 10;

   FUNCTION func_aftertax(sal NUMBER) RETURN NUMBER
      IS
         tax NUMBER := 0.05;
      BEGIN
         RETURN (ROUND(sal - (sal * tax)));
   END func_aftertax;

   PROCEDURE pro_emp(in_empno IN EMP.EMPNO%TYPE)
      IS
         out_ename EMP.ENAME%TYPE;
         out_sal EMP.SAL%TYPE;
      BEGIN
         SELECT ENAME, SAL INTO out_ename, out_sal
           FROM EMP
          WHERE EMPNO = in_empno;

         DBMS_OUTPUT.PUT_LINE('ENAME : ' || out_ename);
         DBMS_OUTPUT.PUT_LINE('SAL : ' || out_sal);
   END pro_emp;

PROCEDURE pro_dept(in_deptno IN DEPT.DEPTNO%TYPE)
   IS
      out_dname DEPT.DNAME%TYPE;
      out_loc DEPT.LOC%TYPE;
   BEGIN
      SELECT DNAME, LOC INTO out_dname, out_loc
        FROM DEPT
       WHERE DEPTNO = in_deptno;

      DBMS_OUTPUT.PUT_LINE('DNAME : ' || out_dname);
      DBMS_OUTPUT.PUT_LINE('LOC : ' || out_loc);
   END pro_dept;
END;
/
```

Q028

- 다음과 같이 프로시저 오버로드 하시오.

  Package PKG_OVERLOAD이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PACKAGE pkg_overload
IS
   PROCEDURE pro_emp(in_empno IN EMP.EMPNO%TYPE);
   PROCEDURE pro_emp(in_ename IN EMP.ENAME%TYPE);
END;
/
```

Q029

- 패키지 본문에서 오버로드된 프로시저를 작성하시오.

  Package Body PKG_OVERLOAD이(가) 컴파일되었습니다.

```
CREATE OR REPLACE PACKAGE BODY pkg_overload
IS
   PROCEDURE pro_emp(in_empno IN EMP.EMPNO%TYPE)
      IS
         out_ename EMP.ENAME%TYPE;
         out_sal EMP.SAL%TYPE;
      BEGIN
         SELECT ENAME, SAL INTO out_ename, out_sal
           FROM EMP
          WHERE EMPNO = in_empno;

         DBMS_OUTPUT.PUT_LINE('ENAME : ' || out_ename);
         DBMS_OUTPUT.PUT_LINE('SAL : ' || out_sal);
      END pro_emp;

   PROCEDURE pro_emp(in_ename IN EMP.ENAME%TYPE)
      IS
         out_ename EMP.ENAME%TYPE;
         out_sal EMP.SAL%TYPE;
      BEGIN
         SELECT ENAME, SAL INTO out_ename, out_sal
           FROM EMP
          WHERE ENAME = in_ename;

         DBMS_OUTPUT.PUT_LINE('ENAME : ' || out_ename);
         DBMS_OUTPUT.PUT_LINE('SAL : ' || out_sal);
      END pro_emp;

END;
/
```


Q030

- 패키지에 포함된 서브프로그램 실행하시오.

```
--pkg_example.func_aftertax(3000)--
after-tax:2850
--pkg_example.pro_emp(7788)--
ENAME : SCOTT
SAL : 3000
--pkg_example.pro_dept(10)--
DNAME : ACCOUNTING
LOC : NEW YORK
--pkg_overload.pro_emp(7788)--
ENAME : SCOTT
SAL : 3000
--pkg_overload.pro_emp('SCOTT')--
ENAME : SCOTT
SAL : 3000


PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

```
BEGIN
   DBMS_OUTPUT.PUT_LINE('--pkg_example.func_aftertax(3000)--');
   DBMS_OUTPUT.PUT_LINE('after-tax:' || pkg_example.func_aftertax(3000));

   DBMS_OUTPUT.PUT_LINE('--pkg_example.pro_emp(7788)--');
   pkg_example.pro_emp(7788);

   DBMS_OUTPUT.PUT_LINE('--pkg_example.pro_dept(10)--' );
   pkg_example.pro_dept(10);

   DBMS_OUTPUT.PUT_LINE('--pkg_overload.pro_emp(7788)--' );
   pkg_overload.pro_emp(7788);

   DBMS_OUTPUT.PUT_LINE('--pkg_overload.pro_emp(''SCOTT'')--' );
   pkg_overload.pro_emp('SCOTT');
END;
/
```

Q031

- EMP_TRG테이블을 다음과 같이 생성하시오.

  Table EMP_TRG이(가) 생성되었습니다.

```
CREATE TABLE EMP_TRG
    AS SELECT * FROM EMP;
```

Q032

- DML 실행 전에 수행할 트리거를 생성하시오.

  Trigger TRG_EMP_NODML_WEEKEND이(가) 컴파일되었습니다.

```
CREATE OR REPLACE TRIGGER trg_emp_nodml_weekend
BEFORE
INSERT OR UPDATE OR DELETE ON EMP_TRG
BEGIN
   IF TO_CHAR(sysdate, 'DY') IN ('토', '일') THEN
      IF INSERTING THEN
         raise_application_error(-20000, '주말 사원정보 추가 불가');
      ELSIF UPDATING THEN
         raise_application_error(-20001, '주말 사원정보 수정 불가');
      ELSIF DELETING THEN
         raise_application_error(-20002, '주말 사원정보 삭제 불가');
      ELSE
         raise_application_error(-20003, '주말 사원정보 변경 불가');
      END IF;
   END IF;
END;
/
```

Q033

- 평일날짜로 EMP_TRG 테이블을 UPDTE 하시오.
  SQL 오류: ORA-20001: 주말 사원정보 수정 불가
  ORA-06512: at "SCOTT.TRG_EMP_NODML_WEEKEND", line 6
  ORA-04088: error during execution of trigger 'SCOTT.TRG_EMP_NODML_WEEKEND'

  https://docs.oracle.com/error-help/db/ora-20001/

  More Details :
  https://docs.oracle.com/error-help/db/ora-20001/
  https://docs.oracle.com/error-help/db/ora-06512/
  https://docs.oracle.com/error-help/db/ora-04088/

```
UPDATE emp_trg SET sal = 3500 WHERE empno = 7788;
```

Q034

- 주말날짜에 EMP_TRG테이블을 UPDATE 하시오.
  ORA-20001: 주말 사원정보 수정 불가
  ORA-06512: at "SCOTT.TRG_EMP_NODML_WEEKEND", line 6
  ORA-04088: error during execution of trigger 'SCOTT.TRG_EMP_NODML_WEEKEND'

  https://docs.oracle.com/error-help/db/ora-20001/

  More Details :
  https://docs.oracle.com/error-help/db/ora-20001/
  https://docs.oracle.com/error-help/db/ora-06512/
  https://docs.oracle.com/error-help/db/ora-04088/

```
UPDATE emp_trg SET sal = 3500 WHERE empno = 7788;
```

Q035

- EMP-TRG_LOG 테이블을 생성하시오.

  Table EMP_TRG_LOG이(가) 생성되었습니다.

```
CREATE TABLE EMP_TRG_LOG(
   TABLENAME VARCHAR2(10), -- DML이 수행된 테이블 이름
   DML_TYPE VARCHAR2(10),  -- DML 명령어의 종류
   EMPNO NUMBER(4),        -- DML 대상이 된 사원 번호
   USER_NAME VARCHAR2(30), -- DML을 수행한 USER 이름
   CHANGE_DATE DATE        -- DML이 수행된 날짜
);
```

Q036

- DML 실행 후 수행할 트리거를 생성하시오.

  Trigger TRG_EMP_LOG이(가) 컴파일되었습니다.

```
CREATE OR REPLACE TRIGGER trg_emp_log
AFTER
INSERT OR UPDATE OR DELETE ON EMP_TRG
FOR EACH ROW

BEGIN

   IF INSERTING THEN
      INSERT INTO emp_trg_log
      VALUES ('EMP_TRG', 'INSERT', :new.empno,
               SYS_CONTEXT('USERENV', 'SESSION_USER'), sysdate);

   ELSIF UPDATING THEN
      INSERT INTO emp_trg_log
      VALUES ('EMP_TRG', 'UPDATE', :old.empno,
               SYS_CONTEXT('USERENV', 'SESSION_USER'), sysdate);

   ELSIF DELETING THEN
      INSERT INTO emp_trg_log
      VALUES ('EMP_TRG', 'DELETE', :old.empno,
               SYS_CONTEXT('USERENV', 'SESSION_USER'), sysdate);
   END IF;
END;
/
```


Q037

- EMP_TRG 테이블에 INSERT 실행하시오.

```
오류 발생 명령행: 349 열: 13
오류 보고 -
SQL 오류: ORA-20000: 주말 사원정보 추가 불가
ORA-06512: at "SCOTT.TRG_EMP_NODML_WEEKEND", line 4
ORA-04088: error during execution of trigger 'SCOTT.TRG_EMP_NODML_WEEKEND'

https://docs.oracle.com/error-help/db/ora-20000/20000. 00000 -  "%s"
*Cause:    The stored procedure 'raise_application_error'
           was called which causes this error to be generated.
*Action:   Correct the problem as described in the error message or contact
           the application administrator or DBA for more information.

More Details :
https://docs.oracle.com/error-help/db/ora-20000/
https://docs.oracle.com/error-help/db/ora-06512/
https://docs.oracle.com/error-help/db/ora-04088/
```

```sql
INSERT INTO EMP_TRG
VALUES(9999, 'TestEmp', 'CLERK', 7788,
       TO_DATE('2018-03-03', 'YYYY-MM-DD'), 1200, null, 20);
```

Q038

- COMMIT 하시오.
  커밋 완료.

```sql
COMMIT;
```

Q039

- EMP_TRG 테이블의 INSERT를 확인하시오.

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839	KING	PRESIDENT	(null)	81/11/17	5000	(null)	10
2	7698	BLAKE	MANAGER	7839	81/05/01	2850	(null)	30
3	7782	CLARK	MANAGER	7839	81/05/09	2450	(null)	10
4	7566	JONES	MANAGER	7839	81/04/01	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	81/09/10	1250	1400	30
6	7499	ALLEN	SALESMAN	7698	81/02/11	1600	300	30
7	7844	TURNER	SALESMAN	7698	81/08/21	1500	0	30
8	7900	JAMES	CLERK	7698	81/12/11	950	(null)	30
9	7521	WARD	SALESMAN	7698	81/02/23	1250	500	30
10	7902	FORD	ANALYST	7566	81/12/11	3000	(null)	20
11	7369	SMITH	CLERK	7902	80/12/09	800	(null)	20
12	7788	SCOTT	ANALYST	7566	82/12/22	3000	(null)	20
13	7876	ADAMS	CLERK	7788	83/01/15	1100	(null)	20
14	7934	MILLER	CLERK	7782	82/01/11	1300	(null)	10

```
SELECT *
  FROM EMP_TRG;
```

Q040

- EMP_TRG_LOG 테이블의 INSERT를 기록을 확인하시오.

	TABLEN...	DML_TY...	EMPNO	USER_N...	CHANG...

```
SELECT *
  FROM EMP_TRG_LOG;
```

Q041

- EMP-TRG 테이블에 UPDATE 를 실행하시오.

```
명령의 365 행에서 시작하는 중 오류 발생 -
UPDATE EMP_TRG
   SET SAL = 1300
 WHERE MGR = 7788
오류 발생 명령행: 365 열: 8
오류 보고 -
SQL 오류: ORA-20001: 주말 사원정보 수정 불가
ORA-06512: at "SCOTT.TRG_EMP_NODML_WEEKEND", line 6
ORA-04088: error during execution of trigger 'SCOTT.TRG_EMP_NODML_WEEKEND'

https://docs.oracle.com/error-help/db/ora-20001/

More Details :
https://docs.oracle.com/error-help/db/ora-20001/
https://docs.oracle.com/error-help/db/ora-06512/
https://docs.oracle.com/error-help/db/ora-04088/
커밋 완료.
```

```sql
UPDATE EMP_TRG
   SET SAL = 1300
 WHERE MGR = 7788;

COMMIT;
```

Q042

- USER_TRIGGERS 로 트리거 정보를 조회하시오.

	TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_NA
1	TRG_EMP_LOG	AFTER EACH ROW	INSERT OR UPDATE OR DELETE	EMP_TRG
2	TRG_EMP_NODML_WEEKEND	BEFORE STATEMENT	INSERT OR UPDATE OR DELETE	EMP_TRG

```sql
SELECT TRIGGER_NAME, TRIGGER_TYPE, TRIGGERING_EVENT, TABLE_NAME, STATUS
  FROM USER_TRIGGERS;
```

■STEP2. EX

EX001

- 다음의 결과가 나오도록 내용을 작성하시오.

1. DEPT테이블의 부서번호(DEPT_NO)를 입력값으로 받으 후 부서번호(DEPTNO) , 부서이름(DNAME) , 지역(LOC)을 출력하는 pro_dept_in을 작성하시오.
2. pro_dept_in 프로시저를 통해 출력된 부서번호(DEPTNO) , 부서이름(DNAME) , 지역(LOC)을 다음과 같이 출력하는 pl/sql프로그램을 작성하시오.

```
Procedure PRO_DEPT_IN이(가) 컴파일되었습니다.

부서번호 : 10
부서명 : ACCOUNTING
지역 : NEW YORK


PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

EX002

- 다음과 같은 결과가 나오도록 내욜을 작성하시오.

1. select문에서 사용할 수 있는 func_date_kor을 작성하시오.
2. func_date_kor 함수는 date 자료형데이터를 입력받아 다음과 같이 YYYY"년"MM"월"DD"일" 형태의 데이터를 출력하시오.

```
Function FUNC_DATE_KOR이(가) 컴파일되었습니다.
```

EX003

1. DEPT테이블과 같은 열구조 및 데이터를 가진 DEPT_TRG 테이블을 작성하시오.

2. DEPT_TRG 테이블에 DML 명령어를 사용한 기록을 저장하는 DEPT_TRG_LOG 테이블을 다음과 같이 작성하시오.

열이름	자료형	길이	설명
TABLENAME	가변형문자열	10	DML을 수행한 테이블 이름
DML_TYPE	가변형문자열	10	DML명령어 종류
DEPTNO	정수형 숫자	2	DML 대상 부서번호
USER_NAME	가변형문자열	30	DML을 수행한 USER이름
CHANGE_DATE	날짜	-	DML을 수행한 날짜

3. DEPT_TRG 테이블에 DML 명령수행기록을 DEPT_TRG_LOG에 저장하는 트리거 TRG_DEPT_LOG를 작성하시오.

```
Table DEPT_TRG이(가) 생성되었습니다.

Table DEPT_TRG_LOG이(가) 생성되었습니다.

Trigger TRG_DEPT_LOG이(가) 컴파일되었습니다.
```