PLSQL

OBJECTS
PROCEDURE
FUNCTION
TRIGGER
PACKAGE
CURSOR

ALSO CREATE ANONYMOUS BLOCKS

procedure --
set of instruction thatb are coded for specifcic purpose
it is kept in compliled form.
do not return any value by default.
for dml

FUNCTIONS ===
objects used to do some processing and return final output
always return value
for calculations

PACKAGE ===
set of fucntions and proc for specific purpose

TRIGGER ===
objects that execute when some event gets generated.

CURSOR ===
objects that traverses through the data kept in the memory.

ANONYMOUS BLOCKS ===
not retained physically
translated and executed

INDENTATIONS

```
================================================================================
========
DECLARE
A INT:=&D;
B INT:=10;
   BEGIN
      DBMS_OUTPUT.PUT_LINE('addition='||(A+B));

END;
/
================================================================================
========
```

variable that holds value

SYNTAX FOR DEFINING PROCEDURE

```
==========
========
CREATE OR REPLACE PROCEDURE PR_13
AS
   BEGIN
      DBMS_OUTPUT.PUT_LINE('HELLO');
   END;

EXECUTE PR_13;
OR
EXEC PR_13;
```

```
==========
========
CREATE OR REPLACE PROCEDURE PR_13(A INT,B INT)
AS
   BEGIN
      DBMS_OUTPUT.PUT_LINE(A*B);
   END;
/
EXECUTE PR_13(4,5);
```

```
==========
========
SHOW ERROR
```

OR

```
SHOW ERR -- COMMAND TO DISPLAY ERRORS IN PROCEDURE
==========
========
```

CREATE A PROCEDURE TO PERFORM ALL ARITHMETIC OPERATIONS USING GIVEN PARAMETES.

```
CREATE OR REPLACE PROCEDURE PR_13(A INT,B INT)
AS
   BEGIN
            DBMS_OUTPUT.PUT_LINE('MULTIPLICATION - ' || (A*B));
      DBMS_OUTPUT.PUT_LINE('ADDITION - '|| (A+B));
      DBMS_OUTPUT.PUT_LINE('DIVISION - '|| (A/B));
      DBMS_OUTPUT.PUT_LINE('SUBTRACTION - '|| (A-B));
   END;
/
EXECUTE PR_13(4,5);
```

```
================================================================================
========
CREATE OR REPLACE PROCEDURE PR_13(A INT,B INT,C INT)
AS
   BEGIN
          IF(A>B) THEN
          DBMS_OUTPUT.PUT_LINE('a id greater than b');
      ELSIF A>C THEN
          DBMS_OUTPUT.PUT_LINE('A IS GREATER THAN C');
ELSE
DBMS_OUTPUT.PUT_LINE('ELSE PART IS EXE')
       END IF;
   END;
/
EXECUTE PR_13(4,5,8);

================================================================================
========
PROCEDURE TO FIND HIGHEST AND SMALLEST NUMBER FROM 3 VALUES SUPPLIED.

CREATE OR REPLACE PROCEDURE PR_15(A INT,B INT,C INT)
AS
   BEGIN
      IF (A>B) THEN
      IF (A>C) THEN
          DBMS_OUTPUT.PUT_LINE('a id greateST');
      ELSIF (C>B) THEN
          DBMS_OUTPUT.PUT_LINE('C IS GREATEST');
      ELSE THEN
          DBMS_OUTPUT.PUT_LINE('b is greatest');
      END IF;
END IF;
      END IF;
END;




================================================================================
========
CREATE OR REPLACE PROCEDURE PR_13(A INT,B INT,C INT)
AS
   BEGIN
      IF ( A>B AND A>C ) THEN
      DBMS_OUTPUT.PUT_LINE(A||' IS LARGEST NO');
      ELSIF(B>C AND B>A) THEN
          DBMS_OUTPUT.PUT_LINE(B||' IS LARGEST NO');
      ELSE
          DBMS_OUTPUT.PUT_LINE(C||' IS LARGEST NO');
```

```
      END IF;
   END;
/
EXECUTE PR_13(10,20,30);


================================================================================
========
CREATE OR REPLACE PROCEDURE PR_13(A INT,B INT,C INT)
AS
BEGIN
        IF ( A>B AND A>C ) THEN
        DBMS_OUTPUT.PUT_LINE(A||' IS LARGEST NO');
   IF (B>C) THEN
        DBMS_OUTPUT.PUT_LINE(C||' IS SMALLEST NO');
   ELSE
        DBMS_OUTPUT.PUT_LINE(B||' IS SMALLEST NO');
   END IF;

   ELSIF ( B>A AND B>C ) THEN
   DBMS_OUTPUT.PUT_LINE(B||' IS LARGEST NO');
   IF (A>C) THEN
        DBMS_OUTPUT.PUT_LINE(C||' IS SMALLEST NO');
   ELSE
        DBMS_OUTPUT.PUT_LINE(A||' IS SMALLEST NO');
        END IF;

        ELSIF ( C>A AND C>B ) THEN
        DBMS_OUTPUT.PUT_LINE(C||' IS LARGEST NO');
        IF (A>C) THEN
             DBMS_OUTPUT.PUT_LINE(C||' IS SMALLEST NO');
        ELSE
             DBMS_OUTPUT.PUT_LINE(A||' IS SMALLEST NO');
        END IF;
        END IF;
END;



EXECUTE PR_13(10,20,30);
```

```
================================================================================
========
CREATE OR REPLACE PROCEDURE PR_13(A VARCHAR2)
AS
   BEGIN
      IF(A='TECH' or A='tech') THEN
      DBMS_OUTPUT.PUT_LINE('logged in');
else
DBMS_OUTPUT.PUT_LINE('login failed');
end if;
end;



================================================================================
========
CREATE OR REPLACE PROCEDURE PR_13(A INT,B INT DEFAULT 3,C INT DEFAULT 5)
AS
   ANS INT;
BEGIN
ANS:=A+B+C;
DBMS_OUTPUT.PUT_LINE('SUM TOATAL='||ANS);
END;

================================================================================
-------
CREATE OR REPLACE PROCEDURE PR_13(A INT)
AS
 ENAM VARCHAR2(20);
BEGIN
   SELECT ENAME INTO ENAM FROM EMP WHERE EMPNO=A;
   DBMS_OUTPUT.PUT_LINE('NAME='||ENAM||' OF EMPNO'||A);
END;
/

EXEC PR_13(7900);
================================================================================
========

LOOPS

used when we want to get set of statements executed several times.

FIX ITERATION -
FOR LOOP- when we knw no. of times

WHILE - we dont knw no. of times.

CHR(9) - TAB SPACE

================================================================================
```

==========

FOR LOOP - - - - - - - -

```
DECLARE
   BEGIN
      FOR I IN 1..5
      LOOP
      DBMS_OUTPUT.PUT_LINE(I||CHR(9)||CHR(9)||'='||I*I);
      END LOOP;
   END;
```

==========
==========

////////////// DISPLAY TABLE IN FORMAT 5 * 1 = 5

```
DECLARE
A INT:=&A;
BEGIN
   FOR I IN 1..10
      LOOP
         DBMS_OUTPUT.PUT_LINE(A||CHR(9)||'*'||CHR(9)||I||CHR(9)||'='||CHR(9)||A*I);
      END LOOP;
END;
```

==========

////////////// DISPLAY TABLE IN FORMAT 5 * 1 = 5

```
DECLARE
A INT:= &A;
BEGIN
   FOR I IN REVERSE 1..10
      LOOP
         DBMS_OUTPUT.PUT_LINE(A||CHR(9)||'*'||CHR(9)||I||CHR(9)||'='||CHR(9)||A*I);
      END LOOP;
END;
```

==========
==========

////////////// WHILE

```
DECLARE
A INT:= 1;
BEGIN
 WHILE A<=5
       LOOP
          DBMS_OUTPUT.PUT_LINE(A);
       A := A +1;

            END LOOP;
```

```
END;

================================================================================
==============
/////////////// DO WHILE /////////////// EXIT LOOP

DECLARE
A INT:= 1;
BEGIN
      LOOP
        DBMS_OUTPUT.PUT_LINE(A);
      A := A +1;
      EXIT WHEN A>5;
           END LOOP;
END;
================================================================================
==============

%TYPE - -    ASSIGN SAME DATA TYPE TO LOCAL VARIBALE

CREATE OR REPLACE PROCEDURE ABC(ID IN EMP.EMPNO%TYPE)
AS
UNAME EMP.ENAME%TYPE;
BEGIN
SELECT ENAME INTO UNAME FROM EMP WHERE EMPNO=ID;
DBMS_OUTPUT.PUT_LINE('VALUE YOU WANT :' || UNAME);
END;
/

================================================================================
==============
%ROWTYPE - STORE ROW

CREATE OR REPLACE PROCEDURE ABC(ID IN EMP.EMPNO%TYPE)
AS
EDATA EMP%ROWTYPE;
BEGIN
SELECT * INTO EDATA FROM EMP WHERE EMPNO=ID;
DBMS_OUTPUT.PUT_LINE('NAME - ' || EDATA.ENAME);
DBMS_OUTPUT.PUT_LINE('SALARY - ' || EDATA.SAL);
DBMS_OUTPUT.PUT_LINE('HIREDATE - ' || EDATA.HIREDATE);
DBMS_OUTPUT.PUT_LINE('JOB - ' || EDATA.JOB);
END;
/

CREATE OR REPLACE PROCEDURE ABC(ID IN EMP.EMPNO%TYPE)
AS
EDATA EMP%ROWTYPE;
BEGIN
SELECT ENAME INTO EDATA FROM EMP WHERE EMPNO=ID;
DBMS_OUTPUT.PUT_LINE('NAME - ' || EDATA);
```

```
END;
/


================================================================================
===============
CREATE DUMMY EMP;
ACCEPT EMPNO AS PARAM
FETCH DATA N CHECK IF JOB IS CLERK OR SAL
THN INC SAL BY 5% OR ELSE BY 7.5% STORE IN DUMMY
```

```
CREATE OR REPLACE PROCEDURE ABCD(ID IN EMP.EMPNO%TYPE)
AS
EDATA EMP%ROWTYPE;
BEGIN
SELECT * INTO EDATA FROM EMP WHERE EMPNO=ID;
IF (EDATA.JOB = 'SALESMAN' OR EDATA.JOB = 'CLERK')
THEN
      EDATA.SAL :=    EDATA.SAL*.05 + EDATA.SAL;
ELSE
      EDATA.SAL :=    EDATA.SAL*.75 + EDATA.SAL;
END IF;
INSERT     INTO     EMP4     VALUES(EDATA.EMPNO,EDATA.ENAME,EDATA.JOB,EDATA.
MGR,EDATA.HIREDATE,EDATA.SAL,EDATA.COMM,EDATA.DEPTNO)
;
END;
/

================================================================================
===============

CREATE OR REPLACE PROCEDURE ABCD(ID IN EMP.EMPNO%TYPE)
AS
EDATA EMP%ROWTYPE;
BEGIN
SELECT * INTO EDATA FROM EMP WHERE EMPNO=ID;
IF (EDATA.JOB = 'SALESMAN' OR EDATA.JOB = 'CLERK')
THEN
      EDATA.SAL :=    EDATA.SAL*.05 + EDATA.SAL;
ELSE
      EDATA.SAL :=    EDATA.SAL*.75 + EDATA.SAL;
END IF;

UPDATE EMP4 SET SAL=EDATA.SAL WHERE EMPNO=ID;
END;
/

================================================================================
===============
```

```
CREATE OR REPLACE PROCEDURE ABCD(ID IN EMP.EMPNO%TYPE)
AS

NEWSAL EMP4.SAL%TYPE;
EDATA EMP%ROWTYPE;
BEGIN
SELECT * INTO EDATA FROM EMP WHERE EMPNO=ID;
IF (EDATA.JOB = 'SALESMAN' OR EDATA.JOB = 'CLERK')
THEN
      NEWSAL :=    EDATA.SAL*.05 + EDATA.SAL;
ELSE
      NEWSAL :=    EDATA.SAL*.75 + EDATA.SAL;
END IF;

UPDATE EMP4 SET SAL= NEWSAL WHERE EMPNO=ID;
END;
/

================================================================================
================
////////////////////////////// USER DEFINED DATATYPE

DECLARE
TYPE EMPREC IS RECORD
(
ID INT,
ENAM EMP.ENAME%TYPE,
SALARY EMP.SAL%TYPE
);

A1 EMPREC;

ENO INT :=&I;
BEGIN
SELECT EMPNO,ENAME,SAL INTO A1 FROM EMP WHERE EMPNO=ENO;
DBMS_OUTPUT.PUT_LINE(A1.ID||' '||A1.ENAM||' '||A1.SALARY);

END;
================================================================================
================
DECLARE
EDATA EMP4%ROWTYPE;
BEGIN
SELECT * INTO EDATA FROM EMP4 WHERE SAL = (SELECT MAX(SAL) FROM EMP) ;
DBMS_OUTPUT.PUT_LINE('EMP NO = '||EDATA.EMPNO);
DBMS_OUTPUT.PUT_LINE('EMP NAME = '||EDATA.ENAME);
DBMS_OUTPUT.PUT_LINE('EMP JOB = '||EDATA.JOB);
DBMS_OUTPUT.PUT_LINE('EMP HIREDATE = '||EDATA.HIREDATE);
DBMS_OUTPUT.PUT_LINE('EMP SALARY = '||EDATA.SAL);
DBMS_OUTPUT.PUT_LINE('EMP COMM = '||EDATA.COMM);
```

```
DBMS_OUTPUT.PUT_LINE('EMP DEPTNO = '||EDATA.DEPTNO);
END;

================================================================================
================
DECLARE
EDATA EMP4%ROWTYPE;
DDATA DEPT%ROWTYPE;
DNO EMP4.DEPTNO%TYPE;
BEGIN
SELECT * INTO EDATA FROM EMP4 WHERE SAL = (SELECT MAX(SAL) FROM EMP) ;

DNO := EDATA.DEPTNO;

SELECT * INTO DDATA FROM DEPT WHERE DEPTNO=EDATA.DEPTNO;
DBMS_OUTPUT.PUT_LINE('EMP NO = '||EDATA.EMPNO);
DBMS_OUTPUT.PUT_LINE('EMP NAME = '||EDATA.ENAME);
DBMS_OUTPUT.PUT_LINE('EMP JOB = '||EDATA.JOB);
DBMS_OUTPUT.PUT_LINE('EMP HIREDATE = '||EDATA.HIREDATE);
DBMS_OUTPUT.PUT_LINE('EMP SALARY = '||EDATA.SAL);
DBMS_OUTPUT.PUT_LINE('EMP COMM = '||EDATA.COMM);
DBMS_OUTPUT.PUT_LINE('EMP DEPTNO = '||EDATA.DEPTNO);
DBMS_OUTPUT.PUT_LINE('EMP DNAME = '||DDATA.DNAME);
DBMS_OUTPUT.PUT_LINE('EMP DEPTNO = '||DDATA.LOC);

END;

================================================================================
==================

DECLARE

TYPE ETYPE IS RECORD
(
    ENO EMP.EMPNO%TYPE,
    NAME EMP.ENAME%TYPE,
    JOB EMP.JOB%TYPE,
    MGR EMP.MGR%TYPE,
    HIRE EMP.HIREDATE%TYPE,
    SALARY EMP.SAL%TYPE,
    COMM EMP.COMM%TYPE,
    DNO EMP.DEPTNO%TYPE,
    DDNO DEPT.DEPTNO%TYPE,
    DNAME DEPT.DNAME%TYPE,
    DLOC DEPT.LOC%TYPE
);

EDATA ETYPE;

BEGIN
    SELECT * INTO EDATA FROM EMP E JOIN DEPT D ON D.DEPTNO = E.DEPTNO
```

```
WHERE SAL = (SELECT MAX(SAL) FROM EMP) ;
        DBMS_OUTPUT.PUT_LINE('EMP NO = '||EDATA.ENO);
        DBMS_OUTPUT.PUT_LINE('EMP NAME = '||EDATA.NAME);
        DBMS_OUTPUT.PUT_LINE('EMP JOB = '||EDATA.JOB);
        DBMS_OUTPUT.PUT_LINE('EMP HIREDATE = '||EDATA.HIRE);
        DBMS_OUTPUT.PUT_LINE('EMP SALARY = '||EDATA.SALARY);
        DBMS_OUTPUT.PUT_LINE('EMP COMM = '||EDATA.COMM);
        DBMS_OUTPUT.PUT_LINE('EMP DEPTNO = '||EDATA.DNO);
        DBMS_OUTPUT.PUT_LINE('EMP DNAME = '||EDATA.DNAME);
        DBMS_OUTPUT.PUT_LINE('EMP DEPTNO = '||EDATA.DLOC);
END;
/
================================================================================
==================
////////////////////////////////////RETURN VALUE


CREATE OR REPLACE PROCEDURE TR13(ID IN INT,ENAM OUT VARCHAR2)
AS

TYPE ETYPE IS RECORD
(
        JOB EMP.JOB%TYPE,
        SALARY EMP.SAL%TYPE
);

EDATA ETYPE;
BEGIN
SELECT JOB,SAL INTO EDATA FROM EMP WHERE EMPNO=ID;
END;

DECLARE
A VARCHAR2(20);
BEGIN
TR13(7900,A,B);
DBMS_OUTPUT.PUT_LINE('JOB='||A);
DBMS_OUTPUT.PUT_LINE('JOB='||B);
END;

================================================================================
==================

CREATE OR REPLACE PROCEDURE FORMATPHONE(PPHONE IN OUT VARCHAR2)
AS
BEGIN
PPHONE := '('||SUBSTR(PPHONE,1,3)||')'||SUBSTR(PPHONE,4,3)||'-'||SUBSTR(PPHONE,7);
END;

----------> GLOBAL VARIBALE DECLARATION

DECLARE
```

```
BEGIN
:GPHONE := '8103448890'; ----------> ASSIGNING VALUE
END;

EXEC FORMATPHONE(:GPHONE)
================================================================================
==================
SELECT SAL,CASE SAL
WHEN 800 THEN SAL+100
WHEN 1000 THEN SAL+200
ELSE SAL+300
END AS DATA FROM EMP;

================================================================================
==================
SELECT SAL,CASE WHEN JOB='CLERK' THEN SAL+(SAL*0.5)
WHEN JOB='SALESMAN' THEN SAL+(SAL*0.7)
END "BONUS" FROM EMP;

================================================================================
==================
SELECT SAL,CASE
WHEN SAL>1000 THEN SAL*.50
WHEN SAL>800 THEN SAL*.20
END AS DATA FROM EMP;

================================================================================
==================


                    C U R S O R
                    ============
                 /================================\
================================================================================
==================

%FOUND --- CHECK DATA FOUND AFTER FIRING QUERRY
%NOTFOUND --OPPOSITE OF %FOUND
%ISOPEN
%ROWCOUNT


CREATE OR REPLACE PROCEDURE TR_14(ID INT)
AS
BEGIN
UPDATE EMP SET SAL=SAL+100 WHERE EMPNO=ID;
DBMS_OUTPUT.PUT_LINE('NO OF ROW ='||SQL %ROWCOUNT);
END;
```

CURSOR NAME IS SELECT COMMAND

==========
==========
CREATE OR REPLACE PROCEDURE TR_14(JD VARCHAR2)
AS
CURSOR CR_EMP IS SELECT EMPNO,ENAME,SAL FROM EMP WHERE JOB=JD;

INFO CR_EMP%ROWTYPE;

BEGIN

OPEN CR_EMP;
LOOP
FETCH CR_EMP INTO INFO;
EXIT WHEN CR_EMP%NOTFOUND;

DBMS_OUTPUT.PUT_LINE(INFO.EMPNO||' '||INFO.ENAME||' '||INFO.SAL);

END LOOP;
CLOSE CR_EMP;
END;

==========
CREATE OR REPLACE PROCEDURE TR_14(DNO INT)
AS
CURSOR CR_EMP IS SELECT ROWNUM,ENAME,SAL,JOB,COMM FROM EMP WHERE DEPTNO=DNO;
INFO CR_EMP%ROWTYPE;
EDATA DEPT%ROWTYPE;

BEGIN
SELECT * INTO EDATA FROM DEPT WHERE DEPTNO=DNO;

OPEN CR_EMP;
DBMS_OUTPUT.PUT_LINE('DEPTNO:'||EDATA.DEPTNO);
DBMS_OUTPUT.PUT_LINE('NAME:'||EDATA.DNAME);
DBMS_OUTPUT.PUT_LINE('LOCATION:'||EDATA.LOC);
DBMS_OUTPUT.
PUT_LINE('==========');
DBMS_OUTPUT.PUT_LINE('SR.
NO'||CHR(9)||'ENAME'||CHR(9)||'SAL'||CHR(9)||'JOB'||CHR(9)||CHR(9)||'COMM');
DBMS_OUTPUT.
PUT_LINE('==========');
LOOP
FETCH CR_EMP INTO INFO;
EXIT WHEN CR_EMP%NOTFOUND;

```
DBMS_OUTPUT.PUT_LINE(INFO.ROWNUM||CHR(9)||INFO.ENAME||CHR(9)||INFO.
SAL||CHR(9)||INFO.JOB||CHR(9)||INFO.COMM);

END LOOP;
CLOSE CR_EMP;
END;

================================================================================
================
CREATE OR REPLACE PROCEDURE TR_15
AS
CURSOR CR_EMP IS SELECT * FROM EMP WHERE SAL > 1000 AND JOB IN
('MANAGER','CLERK');
INFO CR_EMP%ROWTYPE;

BEGIN
      OPEN CR_EMP;
         LOOP
            FETCH CR_EMP INTO INFO;
            EXIT WHEN CR_EMP%NOTFOUND;
            IF (INFO.JOB='CLERK') THEN
               INSERT INTO TEMP2 VALUES (INFO.EMPNO,INFO.ENAME,INFO.JOB,INFO.
MGR,INFO.HIREDATE,INFO.SAL+INFO.SAL*0.15,                INFO.COMM,INFO.
DEPTNO);
            ELSE
               INSERT INTO TEMP2 VALUES (INFO.EMPNO,INFO.ENAME,INFO.JOB,INFO.
MGR,INFO.HIREDATE,INFO.SAL                +INFO.SAL*0.20,INFO.COMM,INFO.
DEPTNO);
            END IF;
         END LOOP;
      CLOSE CR_EMP;
END;

================================================================================
============================
====================                PARAMETRISED                CURSOR
============================================================
================================================================================
============================

CREATE OR REPLACE PROCEDURE XYZ AS
      CURSOR C_EMP(CIN_NO NUMBER,JOB VARCHAR2) IS
            SELECT ENAME,MGR FROM EMP WHERE EMPNO = CIN_NO;

      V_DEPTNO EMP.EMPNO%TYPE := 7876;
      J    EMP.JOB%TYPE := 'CLERK';
      V_COUNTEMP VARCHAR2(20);
      STATUS NUMBER;

BEGIN
```

```
    OPEN C_EMP(V_DEPTNO,J);

    FETCH C_EMP INTO V_COUNTEMP,STATUS;

    DBMS_OUTPUT.PUT_LINE(V_COUNTEMP);
    DBMS_OUTPUT.PUT_LINE(STATUS);

CLOSE C_EMP;
END;
=============================================================================
=========================
```

1.CREATE A PROC
2.CURSOR THAT WILL FETCH DATA FROM DEPT TABLE
3.DEFINE PARAM CURSOR THAT WILL FETCH DATA DEPT WISE AND DISPLAY SAME

```
CREATE OR REPLACE PROCEDURE XYZ AS
      CURSOR C_DEPT IS SELECT DEPTNO FROM DEPT;
      DNO INT;

      CURSOR C_EMP(CIN_NO INT) IS
            SELECT EMPNO,ENAME,JOB,DEPTNO FROM EMP WHERE DEPTNO= CIN_NO;

            ENO EMP.EMPNO%TYPE;
            DNNO EMP.DEPTNO%TYPE;
            EN EMP.ENAME%TYPE;
            J EMP.JOB%TYPE;
BEGIN
OPEN C_DEPT;
LOOP
FETCH C_DEPT INTO DNO;
EXIT WHEN C_DEPT%NOTFOUND;

OPEN C_EMP(DNO);
LOOP
FETCH C_EMP INTO ENO,EN,J,DNNO;
EXIT WHEN C_EMP%NOTFOUND;

DBMS_OUTPUT.PUT_LINE(ENO||CHR(9)||EN||CHR(9)||J||CHR(9)||DNNO);

END LOOP;
CLOSE C_EMP;
END LOOP;
CLOSE C_DEPT;
END;


=============================================================================
==================================

CREATE OR REPLACE PROCEDURE XYZ AS
```

```
      CURSOR C_DEPT IS SELECT DEPTNO,DNAME FROM DEPT;
      DNO INT;
      DN DEPT.DNAME%TYPE;

      CURSOR C_EMP(CIN_NO INT) IS
            SELECT EMPNO,ENAME,JOB,SAL FROM EMP WHERE DEPTNO= CIN_NO;

            ENO EMP.EMPNO%TYPE;
            SA EMP.SAL%TYPE;
            EN EMP.ENAME%TYPE;
            J EMP.JOB%TYPE;
BEGIN
OPEN C_DEPT;
LOOP
FETCH C_DEPT INTO DNO,DN;
EXIT WHEN C_DEPT%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('DEPARTMENT NO : '||DNO);
DBMS_OUTPUT.PUT_LINE('DEPARTMENT NAME : '||DN);
DBMS_OUTPUT.
PUT_LINE('____________________________________________________________');
DBMS_OUTPUT.
PUT_LINE('EMP_NO'||CHR(9)||'EMP_NAME'||CHR(9)||'EMP_SAL'||CHR(9)||'EMP_JOB');
DBMS_OUTPUT.
PUT_LINE('____________________________________________________________');
OPEN C_EMP(DNO);
LOOP
FETCH C_EMP INTO ENO,EN,J,SA;
EXIT WHEN C_EMP%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(ENO||CHR(9)||EN||CHR(9)||SA||CHR(9)||J);


END LOOP;
CLOSE C_EMP;
DBMS_OUTPUT.
PUT_LINE('______________________________________________________________');
DBMS_OUTPUT.PUT_LINE(' ');
END LOOP;
CLOSE C_DEPT;
END;

==================================================================================================
                ==============================          REF          CURSOR
====================================================
==================================================================================================

DECLARE
TYPE REFT1 IS REF CURSOR RETURN EMP%ROWTYPE;
      EDATA REFT1;
      DATAHOLD EMP%ROWTYPE;
```

```
BEGIN
      OPEN EDATA FOR SELECT * FROM EMP;
            LOOP
                  FETCH EDATA INTO DATAHOLD;
                  EXIT WHEN EDATA%NOTFOUND;
                  DBMS_OUTPUT.PUT_LINE(DATAHOLD.EMPNO||'       '||DATAHOLD.ENAME||'
'||DATAHOLD.SAL);
            END LOOP;
      CLOSE EDATA;
END;
```

==========

==========

1.CREATE PROC
2. CREATE RECORD TYPE WILL HOLD EMPNO,ENAME,SAL,JOB,DNAME AND LOC
3. SIMPLE CURSOR FETCH DETAILS FROM DEPT
4.DATA OF SIMPLE CUR TO REF CUR GET DATA OF EMP WORKING FOR SPEC DEPT AND DESIGNATION AS CLERK

```
CREATE OR REPLACE PROCEDURE XYZ AS

TYPE ETYPE IS RECORD
(
ENO EMP.EMPNO%TYPE,
EN EMP.ENAME%TYPE,
SL EMP.SAL%TYPE,
JB EMP.JOB%TYPE,
DN DEPT.DNAME%TYPE,
LC DEPT.LOC%TYPE
);
      CURSOR C_DEPT IS SELECT * FROM DEPT;
      DDATA DEPT%ROWTYPE;

      TYPE REFT1 IS REF CURSOR RETURN ETYPE;
      EDATA REFT1;
      DATAHOLD ETYPE;

BEGIN
OPEN C_DEPT;
LOOP
FETCH C_DEPT INTO DDATA;
EXIT WHEN C_DEPT%NOTFOUND;

OPEN EDATA FOR SELECT EMPNO,ENAME,SAL,JOB,DNAME,LOC FROM EMP E JOIN
DEPT D ON E.DEPTNO=D.DEPTNO WHERE E.DEPTNO=DDATA.DEPTNO AND
JOB='CLERK';
            LOOP
                  FETCH EDATA INTO DATAHOLD;
```

```
                EXIT WHEN EDATA%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(DATAHOLD.ENO||'                '||DATAHOLD.EN||'
'||DATAHOLD.SL||' '||DATAHOLD.JB||' '||DATAHOLD.DN||' '||DATAHOLD.LC);
            END LOOP;
        CLOSE EDATA;
END LOOP;
CLOSE C_DEPT;
END;
```

```
==========================================================================
==================================
============================    E X C E P T I O N      H A N D L I N G
==================================
==========================================================================
==================================
SYSTEM - - - NAME + ERR CODE
UNNAMED SYS - - -
USER DEFINED

SYS 22 EXCEPTIONS


DECLARE
BEGIN
DBMS_OUTPUT.PUT_LINE(6/0);
EXCEPTION
WHEN ZERO_DIVIDE THEN
DBMS_OUTPUT.PUT_LINE(SQLCODE||' '||SQLERRM);
END;
==========================================================================
==================================
CREATE OR REPLACE PROCEDURE TR13(A INT)
AS
NAME VARCHAR2(20);
BEGIN
SELECT ENAME INTO NAME FROM EMP WHERE EMPNO=A;
DBMS_OUTPUT.PUT_LINE('NAME= '||NAME);
EXCEPTION

WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');

WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('MORE THAN 1 ROW');

WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('SOME OTHER ERROR');
```

```
END;
```

USER DEFINED EXCEPTION
1. DECLARE A VARIBALE OF EXCEPTION TYPE
FOR VIOLATION OF BUSINESS LOGIC

RAISE - - - - GENERATE EXCEPTION

```
DECLARE
INVALID_AGE EXCEPTION;
VID INT;
BEGIN
VID := &UID;
IF(VID<18) THEN
RAISE INVALID_AGE;
ELSE
DBMS_OUTPUT.PUT_LINE('AGE ENTERED IS ='||VID);
END IF;
EXCEPTION WHEN INVALID_AGE THEN
DBMS_OUTPUT.PUT_LINE('AGE SHLD BR GREATER THAN 18');
END;
```

```
CREATE OR REPLACE PROCEDURE TR13(SL EMP.SAL%TYPE)
AS

EDATA EMP%ROWTYPE;


BEGIN

SELECT * INTO EDATA FROM EMP WHERE SAL=SL;
INSERT INTO MSG VALUES(EDATA.ENAME);
EXCEPTION
WHEN NO_DATA_FOUND THEN
INSERT INTO MSG VALUES('No employee with salary of '||SL);

WHEN TOO_MANY_ROWS THEN
INSERT INTO MSG VALUES('More than one employee with salary of '||SL);

WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('SOME OTHER ERROR');
```

```
END;

================================================================================
==================================
DECLARE
TEMP1 INT;
TEMP2 INT;

EDATA1 EMP4%ROWTYPE;
EDATA2 EMP4%ROWTYPE;

EMP1 EMP4.EMPNO%TYPE;
EMP2 EMP4.EMPNO%TYPE;

E_EMP1 EXCEPTION;
E_EMP2 EXCEPTION;
E_EMP3 EXCEPTION;

BEGIN

EMP1 := &E;
EMP2 := &F;

SELECT COUNT(EMPNO) INTO TEMP1 FROM EMP WHERE EMPNO=EMP1 GROUP BY
EMPNO;
SELECT COUNT(EMPNO) INTO TEMP2 FROM EMP WHERE EMPNO=EMP2 GROUP BY
EMPNO;

IF ( TEMP2 = 1 AND TEMP1 = 1 ) THEN
UPDATE EMP4 SET SAL=SAL*1.20 WHERE EMPNO=EMP2;
UPDATE EMP4 SET SAL=SAL*1.10 WHERE EMPNO=EMP1;
END IF;

IF (TEMP1 = 0 AND TEMP2 = 1) THEN
UPDATE EMP4 SET SAL=SAL*1.20 WHERE EMPNO=EMP2;
RAISE E_EMP1;
END IF;

IF (TEMP1 = 1 AND TEMP2 = 0) THEN
UPDATE EMP4 SET SAL=SAL*1.10 WHERE EMPNO=EMP1;
RAISE E_EMP2;
END IF;

EXCEPTION

WHEN E_EMP1 THEN
DBMS_OUTPUT.PUT_LINE('NOT PRES '||EMP1);

WHEN E_EMP2 THEN
DBMS_OUTPUT.PUT_LINE('NOT PRES '||EMP2);
```

```
WHEN E_EMP3 THEN
DBMS_OUTPUT.PUT_LINE('BOTH');

WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');

WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('MORE THAN 1 ROW');

WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('OTHER');

END;

================================================================================
==================================
================================================================================
==================================

CREATE OR REPLACE PROCEDURE CURR_EX(A INT)
AS
CURSOR CUR_PARA(B INT) IS SELECT * FROM EMP WHERE DEPTNO=B;
EDATA EMP%ROWTYPE;
BEGIN
  OPEN CUR_PARA(A);
  LOOP
FETCH CUR_PARA INTO EDATA;
EXIT WHEN CUR_PARA%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(EDATA.ENAME||' '||EDATA.JOB);
END LOOP;
END;

================================================================================
==================================
////////////////////////////////////////////    F    U    N    C    T    I    O    N    S
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
================================================================================
==================================

CREATE OR REPLACE FUNCTION FUNC_EX(A INT)
RETURN INT
AS

BEGIN
DBMS_OUTPUT.PUT_LINE('MY FIRST FUNC');
RETURN A*A;
END;
/

----------------------------------------->    TO RUN
```

```
DECLARE
ANS INT;
BEGIN
ANS := FUNC_EX(9);
DBMS_OUTPUT.PUT_LINE(ANS);
END;
/

----------------------------------------> OR

EXEC DBMS_OUTPUT.PUT_LINE(FUNC_EX(9));

================================================================================
==================================
CREATE OR REPLACE FUNCTION FUNC_EX(A INT)
RETURN VARCHAR2
AS
NAME VARCHAR2(20);
BEGIN
SELECT ENAME INTO NAME FROM EMP WHERE EMPNO=A;
RETURN NAME;
END;
/


DECLARE
ANS VARCHAR2(20);
BEGIN
ANS := FUNC_EX(7900);
DBMS_OUTPUT.PUT_LINE(ANS);
END;
/


EXEC DBMS_OUTPUT.PUT_LINE(FUNC_EX(7900));

================================================================================
==================================
CREATE OR REPLACE FUNCTION TAX(PVAL IN NUMBER)
RETURN NUMBER
AS
NAME VARCHAR2(20);
BEGIN

RETURN (PVAL*0.05);
END TAX;
/


SELECT ENAME,SAL,TAX(SAL) FROM EMP ORDER BY TAX(SAL);
SELECT ENAME,SAL,TAX(SAL) FROM EMP WHERE TAX(SAL) = 55 ;
```

```
INSERT INTO EMP4(EMPNO,ENAME,MGR,DEPTNO,SAL) VALUES
(4545,'JACK',7900,10,TAX(90000))
```

==========

```
CREATE OR REPLACE FUNCTION FUNC_EX(A INT)
RETURN VARCHAR2
AS
NAME VARCHAR2(20);

BEGIN
SELECT ENAME INTO NAME FROM EMP WHERE EMPNO=A;
RETURN NAME;

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA');
RETURN 'EXEC GEN';

END;
/

EXEC DBMS_OUTPUT.PUT_LINE(FUNC_EX(790));
```

==========

- Create a function named USER_ANNUAL_COMP that has three parameters p_eno, p_sal and p_comm for passing on the values of an employee_id, the current salary and commission of the employee respectively. The function calculates and returns the annual compensation of the employee by using the following formula.

  annual_compensation = (p_sal+p_comm)*12

  If the salary or commission value is NULL then zero should be substituted for it.
- Give a call to USER_ANNUAL_COMP from a SELECT statement, against the EMPLOYEES table.

```
CREATE OR REPLACE FUNCTION USER_ANNUAL_COMP(P_ENO INT, P_SAL INT,
P_COMM INT)
RETURN VARCHAR2
AS

EDATA EMP%ROWTYPE;
A_COM NUMBER;
BEGIN
SELECT * INTO EDATA FROM EMP WHERE EMPNO=P_ENO;


IF (P_SAL IS NULL) THEN
```

```
A_COM :=    (P_COMM)*12 ;
ELSIF (P_COMM IS NULL) THEN
A_COM :=    (P_SAL)*12 ;
ELSE
A_COM :=    (P_SAL + P_COMM)*12 ;

END IF;

RETURN A_COM;

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA');
RETURN 'EXEC GEN';

END;
/

EXEC DBMS_OUTPUT.PUT_LINE(USER_ANNUAL_COMP(7900,5000,300));

--------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION USER_ANNUAL_COMP(P_ENO INT)
RETURN VARCHAR2
AS

EDATA EMP%ROWTYPE;
A_COM NUMBER;
BEGIN


SELECT * INTO EDATA FROM EMP WHERE EMPNO=P_ENO;

IF (EDATA.SAL IS NULL) THEN
EDATA.SAL := 0;
ELSIF (EDATA.COMM IS NULL) THEN
EDATA.COMM := 0;
END IF;

A_COM :=    (EDATA.SAL + EDATA.COMM)*12 ;

RETURN A_COM;

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA');
RETURN 'EXEC GEN';

END;
/

EXEC DBMS_OUTPUT.PUT_LINE(USER_ANNUAL_COMP(7900,5000,300));
```

```
SELECT ENAME,USER_ANNUAL_COMP(EMPNO) FROM EMP;
```

==========

```
CREATE OR REPLACE FUNCTION CHK_DATA1(A INT)
RETURN BOOLEAN
AS
CNT INT;
BEGIN
SELECT COUNT(EMPNO) INTO CNT FROM EMP WHERE EMPNO=A;
IF CNT>0 THEN
RETURN TRUE;
ELSE
RETURN FALSE;
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RETURN FALSE;
WHEN OTHERS THEN
RETURN FALSE;
END;
```

```
DECLARE
ANS BOOLEAN;
BEGIN
ANS:=CHK_DATA1(4541);
IF ANS=TRUE THEN
DBMS_OUTPUT.PUT_LINE('RECORD EXIST');
ELSE
DBMS_OUTPUT.PUT_LINE('RECORD DOES NOT EXISTS');
END IF;
END;
```

```
CREATE OR REPLACE PROCEDURE CH(A INT)
AS
ANS BOOLEAN;
BEGIN
ANS:=CHK_DATA1(A);
IF ANS=TRUE THEN
DBMS_OUTPUT.PUT_LINE('RECORD EXIST');
ELSE
DBMS_OUTPUT.PUT_LINE('RECORD DOES NOT EXISTS');
END IF;
END;
```

==========

- Create a function named USER_VALID_DEPTNO that has a single parameter p_dno

to accept a department number and returns a BOOLEAN value. The function returns TRUE if the department number exists in the DEPARTMENTS table else it returns FALSE.

- Create a procedure named SHOW_STRENGTH that accepts department number in a single parameter p_deptno from user. The procedure gives a call to USER_VALID_DEPTNO. If the function returns TRUE then the procedure finds out how many employees are there in the department from the EMPLOYEES table and displays the same on the screen. If the function returns FALSE then the procedure displays an appropriate error message.
- Give call to SHOW_STRENGTH by passing on department number 10. Do the same for department number 76

```
CREATE OR REPLACE FUNCTION USER_VALID_DEPTNO(P_DNO INT)
RETURN BOOLEAN
AS

CNT INT;

BEGIN
   SELECT COUNT(EMPNO) INTO CNT FROM EMP WHERE DEPTNO=P_DNO;
   IF CNT>0 THEN
      RETURN TRUE;
   ELSE
      RETURN FALSE;
   END IF;

   EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN FALSE;
   WHEN OTHERS THEN
      RETURN FALSE;
END;
/

-------------------------------------------------
CREATE OR REPLACE PROCEDURE SHOW_STRENGTH(P_DNO INT)
AS

ANS BOOLEAN;
CNT INT;

BEGIN
   ANS := USER_VALID_DEPTNO(P_DNO);
   IF ANS = TRUE THEN
      SELECT COUNT(EMPNO) INTO CNT FROM EMP WHERE DEPTNO = P_DNO;
      DBMS_OUTPUT.PUT_LINE('NO. OF EMPLOYEES IN DEPT '||P_DNO||' IS '||CNT);
   ELSE
      DBMS_OUTPUT.PUT_LINE('RECORD DOES NOT EXISTS');
   END IF;
END;
```

```
/

---------------------- for displaying record ----------------------------
CREATE OR REPLACE PROCEDURE SHOW_STRENGTH(P_DNO INT)
AS

CURSOR CR_EMP IS SELECT * FROM EMP WHERE DEPTNO = P_DNO;

ANS BOOLEAN;
CNT INT;
EDATA EMP%ROWTYPE;

BEGIN
   ANS := USER_VALID_DEPTNO(P_DNO);
   IF ANS = TRUE THEN
      SELECT COUNT(EMPNO) INTO CNT FROM EMP WHERE DEPTNO = P_DNO;
      DBMS_OUTPUT.PUT_LINE('NO. OF EMPLOYEES IN DEPT '||P_DNO||' IS '||CNT);
      OPEN CR_EMP;
      LOOP
      FETCH CR_EMP INTO EDATA;
      EXIT WHEN CR_EMP%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(EDATA.ENAME);
      END LOOP;
   ELSE
      DBMS_OUTPUT.PUT_LINE('RECORD DOES NOT EXISTS');
   END IF;
END;
/
```

==========

==========

- Create a procedure called USER_QUERY_EMP that accepts three parameters. Parameter p_myeno is of IN parameter mode which provides the employee_id value. The other two parameters p_myjob and p_mysal are of OUT mode. The procedure retrieves the salary and job_id of an employee with the provided employee_id and assigns those to the two OUT parameters respectively. The procedure should handle the error if the employee_id does not exist in the EMPLOYEES table by displaying an appropriate message. Use bind variables for the two OUT Parameters.
- Compile the code, invoke the procedure, and display the salary and job title for employee number 200. Do the same for employee number 120.

```
VARIABLE P_MYJOB VARCHAR2;
VARIABLE P_MYSAL NUMBER;

CREATE OR REPLACE PROCEDURE USER_QUERY_EMP
(
P_MYENO IN INT,
```

```
P_MYJOB OUT VARCHAR2(20),
P_MYSAL OUT NUMBER
)

AS
EDATA EMP%ROWTYPE;
DATA INT;
BEGIN

SELECT * INTO EDATA FROM EMP WHERE EMPNO=P_MYENO;

SELECT JOB,SAL INTO P1,P2 FROM EMP WHERE EMPNO=P_MYENO;

DBMS_OUTPUT.PUT_LINE('1'||EDATA.JOB||EDATA.SAL);
--P_MYJOB := EDATA.JOB;
DBMS_OUTPUT.PUT_LINE('1.1'||P1);
--P_MYSAL := EDATA.SAL;
DBMS_OUTPUT.PUT_LINE('JOB '||P1);

DBMS_OUTPUT.PUT_LINE('SAL '||P2);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');

WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE('MORE THAN 1 ROW');

WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('SOME OTHER ERROR');

END;


EXEC USER_QUERY_EMP(7934,:P_MYJOB,:P_MYSAL);
```