```
============================================================================================
                                    EXPERIMENT NO. 06
============================================================================================

  Author   : Diksha Gupta.
  Roll no.: 06 [27A]
  Date     : 25-NOVEMBER-2022.

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

AIM : To write and execute stored procedures and stored functions using Oracle 11g.

PROBLEM STATEMENT:

Using the relation schemata established in Experiments - 02, 03, and 05, create and execute the mentioned stored functions and stored procedures.

```
*************************************** QUERY-01 ******************************************
Write SQL code to compile and execute a stored procedure - SHOW_EMPLOYEE, to list employee
details for the input variable ENO holding employee number. (Use EMPP Table)
********************************************************************************************
```

```
CREATE OR REPLACE PROCEDURE SHOW_EMPLOYEE ( EMP_NO EMPP.EID%TYPE  ,
                                            V_EMP OUT EMPP%ROWTYPE)
AS

BEGIN
      SELECT * INTO V_EMP FROM EMPP WHERE EID = EMP_NO;
      DBMS_OUTPUT.PUT_LINE('EID =  '||V_EMP.EID ||CHR(10)||'NAME =  '||V_EMP.ENAME ||CHR(10)
      || 'HIREDATE =  '||V_EMP.HIREDATE ||CHR(10)||'DESIGNATION =  '||V_EMP.DESIGNATION
      ||CHR(10)||'SALARY =  '||V_EMP.SALARY);

EXCEPTION
          WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('INVALID DATA');
END ;
/

Procedure created.
```

```
DECLARE
    EMP_NO EMPP.EID%TYPE  := &EMP_NO;
     V_EMP  EMPP%ROWTYPE;
BEGIN
     SHOW_EMPLOYEE( EMP_NO,V_EMP);
END;
/
```

OUTPUT :

```
Enter value for emp_no: 7112
EID =  7112
NAME =  James Washington
HIREDATE =  22-AUG-17
DESIGNATION =  Research Asst.
SALARY =  44600

PL/SQL procedure successfully completed.
```

*************************************** **QUERY-02** **
Write SQL code to compile and execute a stored procedure - ADD_EMPLOYEE, to add a record to EMPP table. Check the existence of the created procedure using USER_OBJECTS view. Use this procedure to insert following records.

7118, **Your Name,** 07-Jul-2020, Teaching Asst., 25000

7119, Atulya Bharat, 03-Aug-2005, Professor, 162000

****

```
CREATE OR REPLACE PROCEDURE ADD_EMPLOYEE (V_EMP_EID  EMPP.EID%TYPE ,
                                          V_EMP_ENAME  EMPP.ENAME%TYPE ,
                                          V_EMP_HIREDATE EMPP.HIREDATE%TYPE ,
                                          V_EMP_DESIGNATION EMPP.DESIGNATION%TYPE ,
                                          V_EMP_SALARY
EMPP.SALARY%TYPE)

AS
BEGIN
     INSERT
     INTO EMPP VALUES(V_EMP_EID , V_EMP_ENAME , V_EMP_HIREDATE , V_EMP_DESIGNATION ,
     V_EMP_SALARY);
     COMMIT;
     DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCCESSFULLY WITH EID '||V_EMP_EID);
```

```
END ;
/


Procedure created.

DECLARE
BEGIN
      ADD_EMPLOYEE(7118,'Diksha Gupta','07-JUL-2020','Teaching Asst.' , 25000);
      ADD_EMPLOYEE(7119,'Atulya Bharat','03-AUG-2005','Professor' , 162000);
END ;
/
```

OUTPUT :

```
ROW INSERTED SUCCESSFULLY WITH EID 7118
ROW INSERTED SUCCESSFULLY WITH EID 7119

PL/SQL procedure successfully completed.
```

```
SELECT *
   FROM EMPP
     WHERE EID = 7118 OR EID = 7119;

       EID ENAME                 HIREDATE  DESIGNATION         SALARY
---------- --------------------- --------- --------------- ----------
      7118 Diksha Gupta          07-JUL-20 Teaching Asst.       25000
      7119 Atulya Bharat         03-AUG-05 Professor           162000
```

*************************************** **QUERY-03** **
Write SQL code to compile and execute the stored procedure
REMOVE_EMPLOYEE, which will remove the employee record(s) from EMPP table when supplied with an input name phrase (entered always as lower case) indicating employee name (use EMPP table). If the matching employee is not found, an appropriate exception should be raised.
****

```
CREATE OR REPLACE PROCEDURE REMOVE_EMPLOYEE(V_EMP_NAME EMPP.ENAME%TYPE)
   AS
         V_DATA NUMBER(2);
   BEGIN
         SELECT COUNT(*) INTO V_DATA FROM EMPP WHERE UPPER(ENAME) = UPPER(V_EMP_NAME);
```

```
            IF(V_DATA > 0) THEN
               DELETE EMPP WHERE UPPER(ENAME) = UPPER(V_EMP_NAME);
               DBMS_OUTPUT.PUT_LINE('ROW DELETED SUCCESSFULLY WITH NAME '||V_EMP_NAME);
               COMMIT;

            ELSE
               RAISE NO_DATA_FOUND;
            END IF;
EXCEPTION
          WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('RECORD DOES NOT EXIST WITH EMPLOYEE NAME AS
            '||V_EMP_NAME);
   END ;
   /

   Procedure created.

       DECLARE
          V_EMP_NAME EMPP.ENAME%TYPE :=&NAME;
       BEGIN
          REMOVE_EMPLOYEE(V_EMP_NAME);
       END ;
       /
```

OUTPUT :

```
Enter value for name: 'ram'
RECORD DOES NOT EXIST WITH EMPLOYEE NAME AS ram

SELECT *
  FROM EMPP
    WHERE ENAME = 'Diksha Gupta';

       EID ENAME                  HIREDATE  DESIGNATION         SALARY
---------- --------------------- --------- --------------- ----------
      7118 Diksha Gupta          07-JUL-20 Teaching Asst.       25000

DECLARE
   V_EMP_NAME EMPP.ENAME%TYPE :=&NAME;
BEGIN
   REMOVE_EMPLOYEE(V_EMP_NAME);
END ; /
```

OUTPUT :

```
Enter value for name: 'diksha gupta'
ROW DELETED SUCCESSFULLY WITH NAME diksha gupta

SELECT *
  FROM EMPP
    WHERE ENAME = 'Diksha Gupta';

no rows selected
```

*************************************** QUERY-04 **
Write SQL code to compile and execute the stored function - CHECK_ITEM that will report status as **1** if items with mentioned P_CODE are present in the inventory, otherwise reports status as **0.** No exceptions to be handled.
****

```
CREATE OR REPLACE FUNCTION CHECK_ITEM(V_P_CODE ITEM.P_CODE%TYPE)
     RETURN NUMBER
AS
       V_VAR NUMBER(1):=0;
BEGIN
       SELECT COUNT(*) INTO V_VAR FROM ITEM WHERE P_CODE = V_P_CODE;
       RETURN V_VAR;
END;
/

 Function created.

SET SERVEROUT.PUT ON;
BEGIN
       IF(CHECK_ITEM('AB112')=1) THEN
          DBMS_OUTPUT.PUT_LINE('ITEM IS PRESENT IN INVENTORY');
       ELSE
         DBMS_OUTPUT.PUT_LINE('ITEM IS NOT PRESENT IN INVENTORY');
       END IF;
END ;
/
```

OUTPUT :

```
ITEM IS PRESENT IN INVENTORY
```

```
SET SERVEROUT.PUT ON;
BEGIN
       IF(CHECK_ITEM('AE112')=1) THEN
          DBMS_OUTPUT.PUT_LINE('ITEM IS PRESENT IN INVENTORY');
       ELSE
         DBMS_OUTPUT.PUT_LINE('ITEM IS NOT PRESENT IN INVENTORY');
       END IF;
END ;
/
```

OUTPUT :

```
ITEM IS NOT PRESENT IN INVENTORY
```

```
*************************************** QUERY-05 ******************************************
Write a SQL code to compile and execute the stored procedure - ADD_ITEM, that will insert an
item in ITEMS table with given particulars - item code, item description, invoice date,
quantity of purchase, minimum quantity, item price and supplier code.
*********************************************************************************************
```

```
CREATE OR REPLACE PROCEDURE ADD_ITEM (V_P_CODE  ITEM.P_CODE%TYPE ,
                                      V_DESCR  ITEM.DESCR%TYPE ,
                                      V_IN_DATE ITEM.IN_DATE%TYPE DEFAULT SYSDATE ,
                                      V_MIN_QTY ITEM.MIN_QTY%TYPE DEFAULT 6 ,
                                      V_QTY ITEM.QTY%TYPE ,
                                      V_PRICE ITEM.PRICE%TYPE ,
                                      V_CODE
ITEM.V_CODE%TYPE)

AS
BEGIN
      INSERT
        INTO ITEM VALUES(V_P_CODE , V_DESCR ,   V_IN_DATE , V_MIN_QTY , V_QTY ,
           V_PRICE , V_CODE );
      COMMIT;
     DBMS_OUTPUT.PUT_LINE('ROW INSERTED SUCCESSFULLY WITH P_CODE '||V_P_CODE);
END ;
/

Procedure created.

BEGIN
      ADD_ITEM('HT15P','NEW_ITEM..56','17-NOV-22',5,43,9.99,NULL);
```

```
        END;
         /
```

OUTPUT :

```
        ROW INSERTED SUCCESSFULLY WITH P_CODE HT15P
```

*************************************** **QUERY-06** **

Write a SQL code to compile and execute the stored procedure - UPDATE_ITEM, that will update particulars (quantity and/or cost) for an item in ITEMS table with given particulars - item code, quantity of purchase, and item price.
Report an error when the said item (to be updated) does not exist in ITEMS table (the NO_DATA_FOUND exception). Use the CHECK_ITEM function created earlier.

****

```
CREATE OR REPLACE PROCEDURE UPDATE_ITEM(V_P_CODE ITEM.P_CODE%TYPE ,
                                        V_PRICE ITEM.PRICE%TYPE ,
                                        V_QTY ITEM.QTY%TYPE)
AS
BEGIN
       IF(CHECK_ITEM(V_P_CODE)=1) THEN
           UPDATE ITEM
              SET PRICE = V_PRICE , QTY = V_QTY
                 WHERE P_CODE = V_P_CODE;
           COMMIT;
           DBMS_OUTPUT.PUT_LINE('ITEM IS SUCCESSFULLY UPDATED WITH P_CODE AS '||V_P_CODE);

        ELSE
           RAISE NO_DATA_FOUND;

        END IF;

 EXCEPTION
       WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.PUT_LINE('ITEM IS NOT PRESENT IN INVENTORY');

END ;
/
Procedure created.
```

```
SELECT *
  FROM ITEM
     WHERE P_CODE = 'RF100';
P_COD DESCR                              IN_DATE      MIN_QTY        QTY      PRICE     V_CODE
----- ------------------------------ --------- ---------- ---------- ---------- ----------
RF100 Rat Tail File                      15-DEC-19         20         43       4.99      21344

BEGIN
      UPDATE_ITEM('RF100',5.99,43);
END;
/
```

OUTPUT :

```
ITEM IS SUCCESSFULLY UPDATED WITH P_CODE AS RF100

PL/SQL procedure successfully completed.
SELECT *
  FROM ITEM
     WHERE P_CODE = 'RF100';

P_COD DESCR                              IN_DATE      MIN_QTY        QTY      PRICE     V_CODE
----- ------------------------------ --------- ---------- ---------- ---------- ----------
RF100 Rat Tail File                      15-DEC-19         20         43       5.99      21344



BEGIN
   UPDATE_ITEM('AF100',3.99,63);
END;
/
```

OUTPUT :

```
ITEM IS NOT PRESENT IN INVENTORY

PL/SQL procedure successfully completed.
```

```
*************************************** QUERY-07 ******************************************
Modify procedure in Query-06, as UPDATE_ITEM_ADD_WHEN_NOT_FOUND such that when the mentioned
item is not present in ITEMS, an item is entered into ITEMS with available particulars
supplied in the procedure call.

The default values for item description, vendor code and minimum quantity as 'NEW ITEM ...',
NULL and (quantity / 8) truncated respectively. Use ADD_ITEM procedure created earlier.
You need not catch the NO_DATA_FOUND exception.
*******************************************************************************************
```

```
CREATE OR REPLACE PROCEDURE UPDATE_ITEM_ADD_WHEN_NO_DATA_FOUND(
                                                  V_P_CODE ITEM.P_CODE%TYPE ,
                                                  V_PRICE ITEM.PRICE%TYPE ,
                                                  V_QTY ITEM.QTY%TYPE)
AS
BEGIN
       IF(CHECK_ITEM(V_P_CODE)=1) THEN
           UPDATE ITEM
               SET PRICE = V_PRICE , QTY = V_QTY
                  WHERE P_CODE = V_P_CODE;
           COMMIT;
           DBMS_OUTPUT.PUT_LINE('ITEM IS SUCCESSFULLY UPDATED WITH P_CODE AS '||V_P_CODE);
       ELSE
           ADD_ITEM(V_P_CODE , 'NEW_ITEM...' , SYSDATE , (V_QTY/8), V_QTY ,V_PRICE ,NULL);
       END IF;
END ;
/
Procedure created.
SELECT * FROM ITEM WHERE P_CODE = 'AF100';
no rows selected

BEGIN
     UPDATE_ITEM_ADD_WHEN_NO_DATA_FOUND('AF100',3.99,63);
END;
/
```

OUTPUT :

```
ROW INSERTED SUCCESSFULLY WITH P_CODE AF100
PL/SQL procedure successfully completed.
SELECT *
    FROM ITEM
       WHERE P_CODE = 'AF100';

P_COD DESCR                          IN_DATE      MIN_QTY        QTY      PRICE     V_CODE
----- ------------------------------ --------- ---------- ---------- ---------- ----------
AF100 NEW_ITEM...                    20-NOV-22          8         63       3.99
```

```
*************************************** QUERY-08 ******************************************
Write a SQL code to compile and execute the stored procedure - SHOW_ITEM that will list the
item particulars for an item in ITEMS table when the item code is supplied as input.
Report an error when the said item to be updated does not exist in ITEMS. Use the CHECK_ITEM
function created earlier.
*********************************************************************************************
```

```
CREATE OR REPLACE PROCEDURE SHOW_ITEM(V_P_CODE ITEM.P_CODE%TYPE)
AS
       V_DATA ITEM%ROWTYPE;
BEGIN
       IF(CHECK_ITEM(V_P_CODE)=1) THEN
           SELECT * INTO V_DATA
               FROM ITEM
                    WHERE P_CODE = V_P_CODE;

           DBMS_OUTPUT.PUT_LINE(V_DATA.P_CODE ||' '||V_DATA.DESCR ||' '||V_DATA.MIN_QTY||'
           '||V_DATA.QTY||' '||V_DATA.IN_DATE||' '||V_DATA.PRICE||' '||V_DATA.V_CODE );
       ELSE
           DBMS_OUTPUT.PUT_LINE('ITEM IS TO BE UPDATED');
       END IF;
END;
/

Procedure created.


DECLARE
       V_P_CODE ITEM.P_CODE%TYPE :=&P_CODE;
BEGIN
       SHOW_ITEM(V_P_CODE);
END;
/
```

OUTPUT :

```
Enter value for p_code: 'CD00X'
CD00X  Cordless Drill  5  12  20-JAN-20  38.95  25595
PL/SQL procedure successfully completed.


Enter value for p_code: 'AF101'
ITEM IS TO BE UPDATED
PL/SQL procedure successfully completed.
```

```
*************************************** QUERY-09 ******************************************
Modify the procedure in Query-08 as SHOW_ITEM_TMR_E which will handle
TOO_MANY_ROWS exception in SELECT query.
In addition to exceptions in Query-06 (NO_DATA_FOUND and OTHERS) the TOO_MANY_ROWS exception
should be caught when a call to the procedure call - EXEC ADD_ITEM('HH15P', 'NEW ITEM-
2',150, NULL, 25); fetches more than one row in
the result set.
*******************************************************************************************
```

```
CREATE OR REPLACE PROCEDURE SHOW_ITEM_TMR_E(V_P_CODE ITEM.P_CODE%TYPE)

AS

       V_DATA ITEM%ROWTYPE;

BEGIN

       SELECT  * INTO V_DATA

           FROM ITEM

              WHERE P_CODE = V_P_CODE ;

       DBMS_OUTPUT.PUT_LINE(RPAD(V_DATA.P_CODE,8) || RPAD(V_DATA.DESCR,20) ||

                             RPAD(V_DATA.IN_DATE,8) || RPAD(V_DATA.MIN_QTY,4)

                             ||RPAD(V_DATA.QTY,8) ||RPAD(V_DATA.PRICE,8) );

EXCEPTION

       WHEN TOO_MANY_ROWS THEN

        DBMS_OUTPUT.PUT_LINE(V_P_CODE ||' MULTIPLE
       ROWS ...................................................................');

END;

/
```

```
Procedure created.
```

```
SET SERVEROUTPUT ON;

BEGIN

       SHOW_ITEM_TMR_E('HH15P');

       SHOW_ITEM_TMR_E('HW15X');

END;

/
```

OUTPUT :

```
HH15P MULTIPLE ROWS ...................................................................

HW15X   Hiveld Hammer       10-JAN-215  60      17.5

PL/SQL procedure successfully completed.
```

```
*************************************** QUERY-10 *****************************************

         Now extend the procedure in Query-09 as SHOW_ITEM_TMR_HANDLED to print the
         rows returned by the SELECT query after catching the appropriate exception
*******************************************************************************************

CREATE OR REPLACE PROCEDURE SHOW_ITEM_TMR_HANDLED(V_P_CODE ITEM.P_CODE%TYPE)
AS
     V_DATA ITEM%ROWTYPE;
BEGIN
          SELECT  * INTO V_DATA
           FROM ITEM
             WHERE P_CODE = V_P_CODE ;

           DBMS_OUTPUT.PUT_LINE(RPAD(V_DATA.P_CODE,8) || RPAD(V_DATA.DESCR,20) ||
            RPAD(V_DATA.IN_DATE,8) || RPAD(V_DATA.MIN_QTY,4) ||RPAD(V_DATA.QTY,8)
            ||RPAD(V_DATA.PRICE,8) );

EXCEPTION
          WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE(V_P_CODE ||' MULTIPLE ROWS
            ...................................................................');

            FOR C IN (SELECT  * INTO V_DATA FROM ITEM WHERE P_CODE = V_P_CODE )
            LOOP

                DBMS_OUTPUT.PUT_LINE(RPAD(C.P_CODE,8) || RPAD(C.DESCR,20) ||
                RPAD(C.IN_DATE,8) || RPAD(C.MIN_QTY,4) ||RPAD(C.QTY,8) ||RPAD(C.PRICE,8) );

            END LOOP;

            DBMS_OUTPUT.PUT_LINE('----------------------------------------------------------
            --------------------------');
END;
/


Procedure created.
```

```
SET SERVEROUTPUT ON;

BEGIN
       SHOW_ITEM_TMR_HANDLED('HH15P');
       SHOW_ITEM_TMR_HANDLED('HW15X');
END;
/
```

OUTPUT :

```
HH15P MULTIPLE ROWS ..............................................................

HH15P   NEW ITEM-2         20-NOV-26   150     25

HH15P   NEW_ITEM...        18-NOV-25   43      4.99

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

HW15X   Hiveld Hammer      10-JAN-215  60      17.5


PL/SQL procedure successfully completed.
```

```
===========================================================================================
                               INFERENCES OF THE EXPERIMENT
===========================================================================================
```

Hence , we have successfully write and execute stored procedures and stored functions using Oracle 11g.