*** EXPERIMENT NO: 07 ***

---

Author : Pradyot Patil

Roll No : 53 [6B]

Date : 5-April-2018

---

**********

QUERY-01: Write a 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)

/*To list the particulars of the SQL Objects – Tables, Indexes, Sequences,Procedures, Functions, Packages, Collections – the views, ALL_OBJECTS,DBA_OBJECTS and USER_OBJECTS may be appropriately used.To recover the SQL code for already created Procedures and Functions useviews - ALL_SOURCE, DBA_SOURCE and USER_SOURCE appropriately.*/

**********

```
CREATE OR REPLACE PROCEDURE SHOW_EMPLOYEE
       (V_EID IN EMPP.EID%TYPE) AS
       P_EID NUMBER;
       P_ROW EMPP%ROWTYPE;
BEGIN
       SELECT *  INTO P_ROW FROM EMPP WHERE EMPP.EID=V_EID;
       DBMS_OUTPUT.PUT_LINE('EMPLOYEE ID :'||P_ROW.EID);
       DBMS_OUTPUT.PUT_LINE('EMPLOYEE NAME :'||P_ROW.ENAME);
       DBMS_OUTPUT.PUT_LINE('EMPLOYEE  HIRE DATE:'||P_ROW.HIREDATE);
       DBMS_OUTPUT.PUT_LINE('EMPLOYEE SALARY :'||P_ROW.SALARY);
EXCEPTION
       WHEN OTHERS THEN
              DBMS_OUTPUT.PUT_LINE('EMPLOYEE ID DOES NOT EXIST');
END;
/

DECLARE
       E_EID NUMBER(4) := &EMP_ID;
BEGIN
       SHOW_EMPLOYEE(E_EID);
       DBMS_OUTPUT.PUT_LINE('DONE.....');
END;
/

Enter value for emp_id: 7109
old   2:        V_EID NUMBER(4) :=(&EMP_ID);
new   2:        V_EID NUMBER(4) :=(7109);
EMPLOYEE ID :7109
EMPLOYEE NAME :Larry Gomes
EMPLOYEE  HIRE DATE:18-MAY-99
EMPLOYEE SALARY :13650
DONE.....

PL/SQL procedure successfully completed.
```

**********

QUERY-02 Write a 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.

7111, Justin Beiber, 01-Jan-2016, 12500

7112, Wilfred Diesel, 02-Feb-2016, 13000.

**********

```
CREATE OR REPLACE PROCEDURE ADD_EMPLOYEE

      (V_EID           EMPP.EID%TYPE,V_NAME           EMPP.ENAME%TYPE,V_DATE
EMPP.HIREDATE%TYPE,V_SALARY EMPP.SALARY%TYPE) AS

BEGIN

      INSERT INTO EMPP VALUES (V_EID,V_NAME,V_DATE,V_SALARY);

      DBMS_OUTPUT.PUT_LINE('INSERTED...');

END;

/

Procedure created.

   SELECT OBJECT_NAME,OBJECT_TYPE,STATUS

   FROM USER_OBJECTS

    WHERE OBJECT_TYPE='PROCEDURE'

   AND OBJECT_NAME='ADD_EMPLOYEE';


OBJECT_NAME       OBJECT_TYPE         STATUS

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

ADD_EMPLOYEEPROCEDURE           VALID


DECLARE

       V_EID EMPP.EID%TYPE := &EID_ID;

     V_NAME EMPP.ENAME%TYPE := &V_NAME;

       V_DATE EMPP.HIREDATE%TYPE := &DATE;

       V_SALARY EMPP.SALARY%TYPE := &V_SAL;

BEGIN

            ADD_EMPLOYEE(V_EID,V_NAME,V_DATE,V_SALARY);
```

```
END;
/
Enter value for eid_id: 7111
old   2:  V_EID EMPP.EID%TYPE := &EID_ID;
new   2:  V_EID EMPP.EID%TYPE := 7111;
Enter value for v_name: 'Justin Beiber'
old   3:      V_NAME EMPP.ENAME%TYPE := &V_NAME;
new   3:      V_NAME EMPP.ENAME%TYPE := 'Justin Beiber';
Enter value for date: '01-JAN-2016'
old   4:  V_DATE EMPP.HIREDATE%TYPE := &DATE;
new   4:  V_DATE EMPP.HIREDATE%TYPE := '01-JAN-2016';
Enter value for v_sal: 12500
old   5:  V_SALARY EMPP.SALARY%TYPE := &V_SAL;
new   5:  V_SALARY EMPP.SALARY%TYPE := 12500;
INSERTED...
PL/SQL procedure successfully completed.

Enter value for eid_id: 7112
old   2:  V_EID EMPP.EID%TYPE := &EID_ID;
new   2:  V_EID EMPP.EID%TYPE := 7112;
Enter value for v_name: 'Wilfred Diesel'
old   3:      V_NAME EMPP.ENAME%TYPE := &V_NAME;
new   3:      V_NAME EMPP.ENAME%TYPE := 'Wilfred Diesel';
Enter value for date: '02-FEB-2016'
old   4:  V_DATE EMPP.HIREDATE%TYPE := &DATE;
new   4:  V_DATE EMPP.HIREDATE%TYPE := '02-FEB-2016';
Enter value for v_sal: 13000
old   5:  V_SALARY EMPP.SALARY%TYPE := &V_SAL;
new   5:  V_SALARY EMPP.SALARY%TYPE := 13000;
INSERTED...

PL/SQL procedure successfully completed
```

**********

QUERY-03 Write a 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 Upper 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_NAME EMPP.ENAME%TYPE) AS
       V_CODE NUMBER;
       MY_EXCECPTION EXCEPTION;
BEGIN
       SELECT COUNT(*) INTO V_CODE FROM EMPP WHERE EMPP.ENAME=V_NAME;
       DBMS_OUTPUT.PUT_LINE(V_CODE);
       IF V_CODE>0 THEN
              DELETE FROM EMPP WHERE EMPP.ENAME=V_NAME;
       ELSE
              RAISE MY_EXCECPTION;
       END IF;
EXCEPTION
       WHEN MY_EXCECPTION THEN
              DBMS_OUTPUT.PUT_LINE('EMPLOYEE NAME NOT FOUND');
END;
/

Procedure created.

DECLARE
       V_NAME EMPP.ENAME%TYPE :='&username';
BEGIN
       REMOVE_EMPLOYEE(V_NAME);
       DBMS_OUTPUT.PUT_LINE('DONE.....');
END;
/
Enter value for username: Wilfred Diesel
old   2: V_NAME EMPP.ENAME%TYPE :='&username';
new   2: V_NAME EMPP.ENAME%TYPE :='Wilfred Diesel';
1
DONE.....

PL/SQL procedure successfully completed.

Enter value for username: WILFRED DIESEL
old   2: V_NAME EMPP.ENAME%TYPE :='&username';
new   2: V_NAME EMPP.ENAME%TYPE :='WILFRED DIESEL';
0
EMPLOYEE NAME NOT FOUND
DONE.....

PL/SQL procedure successfully completed.
```

**********

QUERY-04 Write a 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.

PRE-REQUISITES for Query-04 onwards.. Create a table ITEMS that includes P_CODE, DESCRIPT as DESCR, P_DATE as IN_DATE, P_MIN as MIN_QTY, QTY, P_PRICE as PRICE and V_CODE from the PRODUCT table. ITEMS table must be populated at creation. Modify ITEMS table to add P_CODE as primary key. Also configure the columns IN_DATE and MIN_QTY to assume default values as SYSDATE and 2 respectively.

**********

```
CREATE TABLE ITEM AS
(SELECT P_CODE AS P_CODE, DESCRIPT AS DESCR,P_DATE AS IN_DATE,
       P_MIN AS MIN_QTY,P_PRICE AS PRICE,V_CODE
       FROM PRODUCT
       );

ALTER TABLE ITEM
 MODIFY IN_DATE DEFAULT SYSDATE;

Table altered.

ALTER TABLE ITEM
 MODIFY MIN_QTY DEFAULT 2;

Table altered.

 ALTER TABLE ITEM
 ADD CONSTRAINTS PK_P_CODE PRIMARY KEY (P_CODE);

Table altered.


 SELECT COUNT(*) FROM PRODUCT;

  COUNT(*)
----------
        20

SQL> SELECT COUNT(*) FROM ITEM;

  COUNT(*)
----------
        20


CREATE OR REPLACE FUNCTION CHECK_ITEM
       (FP_CODE ITEMS.P_CODE%TYPE)
RETURN NUMBER AS
       STATUS NUMBER(1);
BEGIN
       STATUS := 0;
       SELECT COUNT(*) INTO STATUS FROM ITEMS WHERE ITEMS.P_CODE=FP_CODE;
       RETURN STATUS;
END;
/
       Function created.
```

```
DECLARE
	V_CODE ITEMS.P_CODE%TYPE := '&item_code';
	V_STATUS NUMBER(1);
BEGIN
	V_STATUS := CHECK_ITEM(V_CODE);
	IF V_STATUS=1 THEN
		DBMS_OUTPUT.PUT_LINE('ITEM FOUND....');
	ELSE
		DBMS_OUTPUT.PUT_LINE('ITEM NOT FOUND....');
	END IF;
END;
/


Enter value for item_code: CD452
old   2: FP_CODE ITEM.P_CODE%TYPE := '&item_code' ;
new   2: FP_CODE ITEM.P_CODE%TYPE := 'CD452' ;
ITEM NOT FOUND.......


Enter value for item_code: JB012
old   2: FP_CODE ITEM.P_CODE%TYPE := '&item_code' ;
new   2: FP_CODE ITEM.P_CODE%TYPE := 'JB012' ;
ITEM FOUND.......


PL/SQL procedure successfully completed.

**********************************************************************************
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_ITEMS
	(I_CODE ITEMS.P_CODE%TYPE , I_DES ITEMS.DESCR%TYPE , I_DATE ITEMS.IN_DATE%TYPE
, I_MINQ ITEMS.MIN_QTY%TYPE , I_PR ITEMS.PRICE%TYPE , I_SUPP ITEMS.V_CODE%TYPE) AS
BEGIN
	INSERT  INTO ITEMS VALUES (I_CODE,I_DES,I_DATE,I_MINQ,I_PR,I_SUPP);
	DBMS_OUTPUT.PUT_LINE('SUCCESSFULLY INSERTED....');
END;
/

Procedure created

DECLARE

	V_CODE ITEMS.P_CODE%TYPE :='&PCODE';
	V_DESCR ITEMS.DESCR%TYPE :='&DESCRIPT';
	V_DATE ITEMS.IN_DATE%TYPE :='&IDATE';
	 V_QTY ITEMS.MIN_QTY%TYPE :=&QTY;
	V_PRICE ITEMS.PRICE%TYPE :=&PRICE;
	V_SUPCODE ITEMS.V_CODE%TYPE := '&VCODE';
BEGIN
		ADD_ITEM(V_CODE,V_DESCR,V_DATE,V_QTY,V_PRICE,V_SUPCODE);
END;
/
```

```
Enter value for pcode: CD452
old   2: V_CODE ITEM.P_CODE%TYPE :='&PCODE';
new   2: V_CODE ITEM.P_CODE%TYPE :='CD452';
Enter value for descript: CHAIN
old   3: V_DESCR ITEM.DESCR%TYPE :='&DESCRIPT';
new   3: V_DESCR ITEM.DESCR%TYPE :='CHAIN';
Enter value for idate: 05-APR-2018
old   4: V_DATE ITEM.IN_DATE%TYPE :='&IDATE';
new   4: V_DATE ITEM.IN_DATE%TYPE :='05-APR-2018';
Enter value for qty: 10
old   5: V_QTY ITEM.MIN_QTY%TYPE :=&QTY;
new   5: V_QTY ITEM.MIN_QTY%TYPE :=10;
Enter value for price: 500
old   6: V_PRICE ITEM.PRICE%TYPE :=&PRICE;
new   6: V_PRICE ITEM.PRICE%TYPE :=500;
Enter value for vcode: 21344
old   7: V_SUPCODE ITEM.V_CODE%TYPE := '&VCODE';
new   7: V_SUPCODE ITEM.V_CODE%TYPE := '21344';

PL/SQL procedure successfully completed.
```

**SELECT * FROM ITEM WHERE P_CODE='AB213';**

```
P_COD DESCR                          IN_DATE      MIN_QTY      PRICE     V_CODE
----- ------------------------------ --------- ---------- ---------- ----------
CD452 CHAIN                          05-APR-18          10        500      21344
```

**********

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_ITEMS
       (I_CODE ITEMS.P_CODE%TYPE , I_MINQ ITEMS.MIN_QTY%TYPE , I_PR ITEMS.PRICE%TYPE)
AS
       MY_EXC EXCEPTION;
       V_CODE NUMBER;
BEGIN
       SELECT COUNT(*) INTO V_CODE FROM ITEMS WHERE P_CODE=I_CODE;
       IF V_CODE= 1 THEN
              UPDATE ITEMS SET MIN_QTY = I_MINQ , PRICE = I_PR ;
              DBMS_OUTPUT.PUT_LINE('UPDATE SUCCESSFUL....');

       ELSE
              RAISE MY_EXC;
       END IF;
EXCEPTION
       WHEN MY_EXC THEN
              DBMS_OUTPUT.PUT_LINE('NO DATA FOUND FOR GIVEN PCODE....');
END;
/

Procedure created.
```

```
DECLARE
	V_CODE ITEMS.P_CODE%TYPE :='&P_CODE';
	V_QTY ITEMS.MIN_QTY%TYPE := &QTY;
	V_PRICE ITEMS.PRICE%TYPE := &PRICE;
BEGIN
		UPDATE_ITEMS(V_CODE,V_QTY,V_PRICE);
END;
/
Enter value for pcode: CD452
old   2: C_CODE ITEM.P_CODE%TYPE :='&PCODE';
new   2: C_CODE ITEM.P_CODE%TYPE :='CD452';
Enter value for qty: 5
old   3:  V_QTY ITEM.MIN_QTY%TYPE := &QTY;
new   3:  V_QTY ITEM.MIN_QTY%TYPE := 5;
Enter value for price: 200
old   4: V_PRICE ITEM.PRICE%TYPE := &PRICE;
new   4: V_PRICE ITEM.PRICE%TYPE := 200;
UPDATED...

PL/SQL procedure successfully completed.

Enter value for pcode: CD000
old   2: C_CODE ITEM.P_CODE%TYPE :='&PCODE';
new   2: C_CODE ITEM.P_CODE%TYPE :='CD000';
Enter value for qty: 5
old   3:  V_QTY ITEM.MIN_QTY%TYPE := &QTY;
new   3:  V_QTY ITEM.MIN_QTY%TYPE := 5;
Enter value for price: 50
old   4: V_PRICE ITEM.PRICE%TYPE := &PRICE;
new   4: V_PRICE ITEM.PRICE%TYPE := 50;
NO DATA FOUND FOR GIVEN P_CODE....


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_NOT_FOUND
	(I_CODE ITEMS.P_CODE%TYPE , I_MINQ ITEMS.MIN_QTY%TYPE , I_PR ITEMS.PRICE%TYPE)
		AS
	MY_EXC EXCEPTION;
	V_CODE NUMBER;
BEGIN
	SELECT COUNT(*) INTO V_CODE FROM ITEMS WHERE P_CODE=I_CODE;
	IF V_CODE= 1 THEN
		UPDATE ITEMS SET MIN_QTY = I_MINQ , PRICE = I_PR ;
		DBMS_OUTPUT.PUT_LINE('UPDATE SUCCESSFUL....');

	ELSE
		RAISE MY_EXC;
	END IF;
EXCEPTION
	WHEN MY_EXC THEN
```

```
            INSERT INTO ITEMS VALUES (I_CODE,'NEW
                  ITEM...',TO_CHAR(SYSDATE),I_MINQ,I_PR,NULL);
            DBMS_OUTPUT.PUT_LINE('ADDED SUCCESSFUL....');

END;
/

Procedure created.
DECLARE
      C_CODE ITEMS.P_CODE%TYPE :='&PCODE';
       V_QTY ITEMS.MIN_QTY%TYPE := &QTY;
      V_PRICE ITEMS.PRICE%TYPE := &PRICE;
BEGIN
            UPDATE_ITEM_ADD_WHEN_NOT_FOUND(C_CODE,V_QTY,V_PRICE);
END;
/

Enter value for pcode: CD000
old   2: C_CODE ITEM.P_CODE%TYPE :='&PCODE';
new   2: C_CODE ITEM.P_CODE%TYPE :='CD000';
Enter value for qty: 5
old   3:  V_QTY ITEM.MIN_QTY%TYPE := &QTY;
new   3:  V_QTY ITEM.MIN_QTY%TYPE := 5;
Enter value for price: 500
old   4: V_PRICE ITEM.PRICE%TYPE := &PRICE;
new   4: V_PRICE ITEM.PRICE%TYPE := 500;
ADDED SUCCESSFULLY....

PL/SQL procedure successfully completed
```

**********

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
      (I_CODE ITEMS.P_CODE%TYPE) AS
      V_CODE NUMBER;
      R ITEMS%ROWTYPE;
BEGIN
      V_CODE := CHECK_ITEM(I_CODE);
      IF V_CODE=1 THEN
            SELECT * INTO R FROM ITEMS WHERE ITEMS.P_CODE=I_CODE;
            DBMS_OUTPUT.PUT_LINE(R.P_CODE||' '||R.DESCR||'          '||R.IN_DATE||'
                  '||R.MIN_QTY||'  '||R.PRICE||'   '||R.V_CODE );
      ELSE
            DBMS_OUTPUT.PUT_LINE('Error in
                  Select.................................................');
      END IF;
END;
/

Procedure created.
```

```
BEGIN
SHOW_ITEM('HH15P');
SHOW_ITEM('HH15X');
SHOW_ITEM('SH200');
SHOW_ITEM('SHU00');
SHOW_ITEM('SH100');
SHOW_ITEM('MP100');
END;
 /
```

```
HH15P Error in Select.....................................................
HH15X Hanging Hook 15in 10-JAN-13 25 200 5.75 24992
SH200 Sledge Hammer     02-JUN-12 3  10  25.8
SHU00 No Data Found .....................................................
SH100 Sledge Hammer     02-JAN-12 5  8   14.4
MP100 No Data Found ..................................................... P

PL/SQL procedure successfully completed.

**********************************END**********************************
```