**\*\*\* EXPERIMENT NO: 06 \*\*\***

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

**Author: Atharva Paliwal**

**Roll No: 40 [5B]**

**Date: 06-November-2020**

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

**AIM:** To write and execute stored procedures and 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**

**(ENO EMPP.EID%TYPE,EMP\_REC IN OUT EMPP%ROWTYPE)**

**AS**

**BEGIN**

**SELECT \* INTO EMP\_REC FROM EMPP WHERE EID = ENO;**

**END;**

**/**

Procedure created.

**DECLARE**

**EMP\_REC EMPP%ROWTYPE;**

**BEGIN**

**SHOW\_EMPLOYEE(&ENO,EMP\_REC);**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE INFO: '||EMP\_REC.EID||' '||EMP\_REC.ENAME**

**||' '||EMP\_REC.HIREDATE||' '||EMP\_REC.SALARY);**

**END;**

**/**

Enter value for eno: 7101

old 4: SHOW\_EMPLOYEE(&ENO,EMP\_REC);

new 4: SHOW\_EMPLOYEE(7101,EMP\_REC);

EMPLOYEE INFO: 7101 Eugene Sabatini 10-OCT-06 150000

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.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CREATE OR REPLACE PROCEDURE ADD\_EMPLOYEE**

**(EID EMPP.EID%TYPE,ENAME EMPP.ENAME%TYPE,HIREDATE EMPP.HIREDATE%TYPE,DESIGNATION**

**EMPP.DESIGNATION%TYPE,SALARY EMPP.SALARY%TYPE)**

**AS**

**BEGIN**

**INSERT INTO EMPP VALUES(EID,ENAME,HIREDATE,DESIGNATION,SALARY);**

**END;**

**/**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_TYPE, CREATED FROM USER\_OBJECTS**

**WHERE OBJECT\_TYPE='PROCEDURE';**

OBJECT\_NAME OBJECT\_TYPE CREATED

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

ADD\_EMPLOYEE PROCEDURE 05-NOV-20

SHOW\_EMPLOYEE PROCEDURE 05-NOV-20

**SELECT COUNT(\*) FROM EMPP;**

COUNT(\*)

----------

17

**BEGIN**

**ADD\_EMPLOYEE(&EID,'&ENAME','&HIREDATE','&DESIGNATION',&SALARY);**

**END;**

Enter value for eid: 7118

Enter value for ename: Atharva Paliwal

Enter value for hiredate: 07-Jul-2020

Enter value for designation: Teaching Asst.

Enter value for salary: 25000

old 2: ADD\_EMPLOYEE(&EID,'&ENAME','&HIREDATE','&DESIGNATION',&SALARY);

new 2: ADD\_EMPLOYEE(7118,'Atharva Paliwal','07-Jul-2020','Teaching Asst.',25000);

**SQL> /**

Enter value for eid: 7119

Enter value for ename: Atulya Bharat

Enter value for hiredate: 03-Aug-2005

Enter value for designation: Professor

Enter value for salary: 162000

old 2: ADD\_EMPLOYEE(&EID,'&ENAME','&HIREDATE','&DESIGNATION',&SALARY);

new 2: ADD\_EMPLOYEE(7119,'Atulya Bharat','03-Aug-2005','Professor',162000);

PL/SQL procedure successfully completed.

**SELECT COUNT(\*) FROM EMPP;**

COUNT(\*)

----------

19

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**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**

**(NAME EMPP.ENAME%TYPE)**

**AS**

**BEGIN**

**DELETE FROM EMPP WHERE LOWER(ENAME) = NAME;**

**IF SQL%NOTFOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('RECORD DELETED');**

**END IF;**

**END;**

**/**

Procedure created.

**BEGIN**

**REMOVE\_EMPLOYEE('&NAME');**

**END;**

**/**

Enter value for name: atulya bharat

old 2: REMOVE\_EMPLOYEE('&NAME');

new 2: REMOVE\_EMPLOYEE('atulya bharat');

RECORD DELETED

PL/SQL procedure successfully completed.

**SQL> /**

Enter value for name: ritika mehta

old 2: REMOVE\_EMPLOYEE('&NAME');

new 2: REMOVE\_EMPLOYEE('ritika mehta');

RECORD NOT FOUND

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**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 TABLE ITEMS**

**AS**

**SELECT P\_CODE,DESCRIPT AS DESCR ,P\_DATE AS IN\_DATE,P\_MIN AS MIN\_QTY,QTY,P\_PRICE AS**

**PRICE,V\_CODE**

**FROM PRODUCT;**

Table created.

**ALTER TABLE ITEMS**

**ADD CONSTRAINT ITEMS\_PK\_P\_CODE PRIMARY KEY (P\_CODE);**

Table altered.

**ALTER TABLE ITEMS**

**MODIFY MIN\_QTY DEFAULT 2;**

Table altered.

**ALTER TABLE ITEMS**

**MODIFY IN\_DATE DEFAULT SYSDATE;**

Table altered.

**SELECT CONSTRAINT\_NAME,CONSTRAINT\_TYPE FROM USER\_CONSTRAINTS**

**WHERE TABLE\_NAME ='ITEMS';**

CONSTRAINT\_NAME C

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

SYS\_C0011927 C

SYS\_C0011928 C

SYS\_C0011929 C

SYS\_C0011930 C

SYS\_C0011931 C

SYS\_C0011932 C

ITEMS\_PK\_P\_CODE P

7 rows selected.

**CREATE OR REPLACE FUNCTION CHECK\_ITEM(INPUT\_CODE ITEMS.P\_CODE%TYPE)**

**RETURN NUMBER**

**AS**

**STATUS NUMBER(2) :=0;**

**BEGIN**

**SELECT COUNT(\*) INTO STATUS FROM ITEMS WHERE P\_CODE = INPUT\_CODE;**

**IF STATUS > 0 THEN**

**RETURN 1;**

**ELSE**

**RETURN 0;**

**END IF;**

**END;**

**/**

Function created.

**BEGIN**

**IF CHECK\_ITEM('&INPUT\_CODE') = 1 THEN**

**DBMS\_OUTPUT.PUT\_LINE('RECORD FOUND');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('RECORD NOT FOUND');**

**END IF;**

**END;**

**/**

Enter value for input\_code: HC100

old 2: IF CHECK\_ITEM('&INPUT\_CODE') = 1 THEN

new 2: IF CHECK\_ITEM('HC100') = 1 THEN

RECORD FOUND

PL/SQL procedure successfully completed.

**SQL> /**

Enter value for input\_code: PP100

old 2: IF CHECK\_ITEM('&INPUT\_CODE') = 1 THEN

new 2: IF CHECK\_ITEM('PP100') = 1 THEN

RECORD NOT 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\_ITEM**

**(P\_CODE ITEMS.P\_CODE%TYPE, DESCR ITEMS.DESCR%TYPE, IN\_DATE ITEMS.IN\_DATE%TYPE, QTY**

**ITEMS.QTY%TYPE, MIN\_QTY ITEMS.MIN\_QTY%TYPE, PRICE ITEMS.PRICE%TYPE, V\_CODE**

**ITEMS.V\_CODE%TYPE)**

**AS**

**BEGIN**

**INSERT INTO ITEMS VALUES**

**(P\_CODE,DESCR,IN\_DATE,QTY,MIN\_QTY,PRICE,V\_CODE);**

**END;**

**/**

Procedure created.

**SELECT COUNT(\*) FROM ITEMS;**

COUNT(\*)

----------

22

**BEGIN**

**ADD\_ITEM('&P\_CODE','&DESCR','&IN\_DATE',&QTY,&MIN\_QTY,&PRICE,&V\_CODE);**

**END;**

**/**

Enter value for p\_code: RD304

Enter value for descr: Rat Tail File

Enter value for in\_date: 29-SEP-20

Enter value for qty: 10

Enter value for min\_qty: 2

Enter value for price: 98.02

Enter value for v\_code: 23119

old 2: ADD\_ITEM('&P\_CODE','&DESCR','&IN\_DATE',&QTY,&MIN\_QTY,&PRICE,&V\_CODE);

new 2: ADD\_ITEM('RD304','Rat Tail File','29-SEP-20',10,2,98.02,23119);

PL/SQL procedure successfully completed.

**SELECT COUNT(\*) FROM ITEMS;**

COUNT(\*)

----------

23

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**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**

**(IN\_P\_CODE ITEMS.P\_CODE%TYPE, IN\_QTY ITEMS.QTY%TYPE, IN\_PRICE ITEMS.PRICE%TYPE)**

**AS**

**BEGIN**

**IF(CHECK\_ITEM(IN\_P\_CODE)=1) THEN**

**UPDATE ITEMS**

**SET QTY = IN\_QTY,**

**PRICE = IN\_PRICE**

**WHERE P\_CODE = IN\_P\_CODE;**

**DBMS\_OUTPUT.PUT\_LINE('Data updated');**

**ELSIF(CHECK\_ITEM(IN\_P\_CODE)=0) THEN**

**RAISE NO\_DATA\_FOUND;**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('No Data Found');**

**END;**

**/**

Procedure created.

**SELECT QTY,PRICE FROM ITEMS WHERE P\_CODE='RD304';**

QTY PRICE

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

2 98.02

**BEGIN**

**UPDATE\_ITEM ('&IN\_P\_CODE',&IN\_QTY,&IN\_PRICE);**

**END;**

**/**

Enter value for in\_p\_code: RD304

Enter value for in\_qty: 5

Enter value for in\_price: 100

old 2: UPDATE\_ITEM ('&IN\_P\_CODE',&IN\_QTY,&IN\_PRICE);

new 2: UPDATE\_ITEM ('RD304',5,100);

Data updated

PL/SQL procedure successfully completed.

**SELECT QTY,PRICE FROM ITEMS WHERE P\_CODE='RD304';**

QTY PRICE

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

5 100

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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**

**(IN\_P\_CODE ITEMS.P\_CODE%TYPE, IN\_QTY ITEMS.QTY%TYPE, IN\_PRICE ITEMS.PRICE%TYPE)**

**AS**

**BEGIN**

**IF(CHECK\_ITEM(IN\_P\_CODE)=1) THEN**

**UPDATE ITEMS**

**SET QTY = IN\_QTY,**

**PRICE = IN\_PRICE**

**WHERE P\_CODE = IN\_P\_CODE;**

**DBMS\_OUTPUT.PUT\_LINE('Data updated');**

**ELSIF(CHECK\_ITEM(IN\_P\_CODE)=0) THEN**

**ADD\_ITEM(IN\_P\_CODE,'NEW ITEM',SYSDATE,IN\_QTY/8,IN\_QTY,IN\_PRICE,NULL);**

**DBMS\_OUTPUT.PUT\_LINE('New Item Added');**

**END IF;**

**END;**

**/**

Procedure created.

**BEGIN**

**UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND ('&IN\_P\_CODE',&IN\_QTY,&IN\_PRICE);**

**END;**

**/**

Enter value for in\_p\_code: RD200

Enter value for in\_qty: 1

Enter value for in\_price: 200

old 2: UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND ('&IN\_P\_CODE',&IN\_QTY,&IN\_PRICE);

new 2: UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND ('RD200',1,200);

New Item Added

PL/SQL procedure successfully completed.

**SELECT \* FROM ITEMS WHERE P\_CODE = 'RD200';**

P\_COD DESCR IN\_DATE MIN\_QTY QTY PRICE V\_CODE

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

RD200 NEW ITEM 05-NOV-20 0 1 200

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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**

**(IN\_P\_CODE ITEMS.P\_CODE%TYPE, ITEM\_REC IN OUT ITEMS%ROWTYPE)**

**AS**

**BEGIN**

**IF(CHECK\_ITEM(IN\_P\_CODE)=1) THEN**

**SELECT \* INTO ITEM\_REC FROM ITEMS WHERE P\_CODE = IN\_P\_CODE;**

**DBMS\_OUTPUT.PUT\_LINE('ITEM INFO: '||ITEM\_REC.P\_CODE||' '||ITEM\_REC.DESCR**

**||' '||ITEM\_REC.IN\_DATE||' '||ITEM\_REC.MIN\_QTY||' '||**

**ITEM\_REC.QTY||' '||ITEM\_REC.PRICE||' '||ITEM\_REC.V\_CODE);**

**ELSIF (CHECK\_ITEM(IN\_P\_CODE)=0) THEN**

**DBMS\_OUTPUT.PUT\_LINE('No Data Found');**

**END IF;**

**END;**

**/**

Procedure created.

**DECLARE**

**ITEM\_REC ITEMS%ROWTYPE;**

**BEGIN**

**SHOW\_ITEM('&IN\_P\_CODE',ITEM\_REC);**

**END;**

**/**

Enter value for in\_p\_code: SM48X

old 4: SHOW\_ITEM('&IN\_P\_CODE',ITEM\_REC);

new 4: SHOW\_ITEM('SM48X',ITEM\_REC);

ITEM INFO: SM48X Steel Malting Mesh 17-JAN-20 5 18 62.95 25595

PL/SQL procedure successfully completed.

SQL> /

Enter value for in\_p\_code: SM40X

old 4: SHOW\_ITEM('&IN\_P\_CODE',ITEM\_REC);

new 4: SHOW\_ITEM('SM40X',ITEM\_REC);

No Data Found

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

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Before executing Query 9**

**ALTER TABLE ITEMS DROP PRIMARY KEY;**

Table altered.

**EXEC ADD\_ITEM('HH15P','NEW ITEM-2',SYSDATE,150,TRUNC(150/8),25,NULL);**

PL/SQL procedure successfully completed.

**SELECT \* FROM ITEMS WHERE P\_CODE = 'HH15P' AND DESCR = 'NEW ITEM-2';**

P\_COD DESCR IN\_DATE MIN\_QTY QTY PRICE V\_CODE

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

HH15P NEW ITEM-2 05-NOV-20 150 18 25

**CREATE OR REPLACE PROCEDURE SHOW\_ITEM\_TMR\_E (**

**ITEM\_CODE ITEMS.P\_CODE%TYPE)**

**IS**

**CNT NUMBER(2);**

**IRECORD ITEMS%ROWTYPE;**

**TOO\_MANY\_ROWS EXCEPTION;**

**NO\_ROW EXCEPTION;**

**BEGIN**

**IF CHECK\_ITEM(ITEM\_CODE)=0 THEN**

**RAISE NO\_ROW;**

**ELSE**

**SELECT COUNT(\*) INTO CNT FROM ITEMS WHERE P\_CODE=ITEM\_CODE;**

**IF CNT>1 THEN**

**RAISE TOO\_MANY\_ROWS;**

**ELSIF CNT=1 THEN**

**SELECT \* INTO IRECORD FROM ITEMS WHERE P\_CODE=ITEM\_CODE;**

**DBMS\_OUTPUT.PUT\_LINE ('P\_CODE: '||IRECORD.P\_CODE);**

**DBMS\_OUTPUT.PUT\_LINE ('DESCRIPT: '||IRECORD.DESCR);**

**DBMS\_OUTPUT.PUT\_LINE ('IN\_DATE: '||IRECORD.IN\_DATE);**

**DBMS\_OUTPUT.PUT\_LINE ('MIN\_QTY: '||IRECORD.MIN\_QTY);**

**DBMS\_OUTPUT.PUT\_LINE ('QTY: '||IRECORD.QTY);**

**DBMS\_OUTPUT.PUT\_LINE ('PRICE: '||IRECORD.PRICE);**

**DBMS\_OUTPUT.PUT\_LINE ('V\_CODE: '||IRECORD.V\_CODE);**

**END IF;**

**END IF;**

**EXCEPTION**

**WHEN TOO\_MANY\_ROWS THEN**

**DBMS\_OUTPUT.PUT\_LINE ('MULTIPLE RECORDS..................');**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE ('INVALID ITEM\_CODE');**

**WHEN NO\_ROW THEN**

**DBMS\_OUTPUT.PUT\_LINE ('ITEM IS NOT PRESENT ');**

**END;**

**/**

Procedure created.

**BEGIN**

**SHOW\_ITEM\_TMR\_E('HH15P');**

**SHOW\_ITEM\_TMR\_E('HH15X');**

**END;**

**/**

P\_CODE: HH15P

DESCRIPT: NEW ITEM-2

IN\_DATE: 05-NOV-20

MIN\_QTY: 150

QTY: 18

PRICE: 25

V\_CODE:

ITEM IS NOT PRESENT

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 (**

**ITEM\_CODE ITEMS.P\_CODE%TYPE)**

**IS**

**CNT NUMBER(2);**

**IRECORD ITEMS%ROWTYPE;**

**TOO\_MANY\_ROWS EXCEPTION;**

**NO\_ROW EXCEPTION;**

**IRECORE ITEMS%ROWTYPE;**

**BEGIN**

**IF CHECK\_ITEM(ITEM\_CODE)=0 THEN**

**RAISE NO\_ROW;**

**ELSE**

**SELECT COUNT(\*) INTO CNT FROM ITEMS WHERE P\_CODE=ITEM\_CODE;**

**IF CNT>1 THEN**

**RAISE TOO\_MANY\_ROWS;**

**ELSIF CNT=1 THEN**

**SELECT \* INTO IRECORD FROM ITEMS WHERE P\_CODE=ITEM\_CODE;**

**DBMS\_OUTPUT.PUT\_LINE ('P\_CODE: '||IRECORD.P\_CODE);**

**DBMS\_OUTPUT.PUT\_LINE ('DESCRIPT: '||IRECORD.DESCR);**

**DBMS\_OUTPUT.PUT\_LINE ('IN\_DATE: '||IRECORD.IN\_DATE);**

**DBMS\_OUTPUT.PUT\_LINE ('MIN\_QTY: '||IRECORD.MIN\_QTY);**

**DBMS\_OUTPUT.PUT\_LINE ('QTY: '||IRECORD.QTY);**

**DBMS\_OUTPUT.PUT\_LINE ('PRICE: '||IRECORD.PRICE);**

**DBMS\_OUTPUT.PUT\_LINE ('V\_CODE: '||IRECORD.V\_CODE);**

**END IF;**

**END IF;**

**EXCEPTION**

**WHEN TOO\_MANY\_ROWS THEN**

**DBMS\_OUTPUT.PUT\_LINE ('MULTIPLE RECORDS..................');**

**DBMS\_OUTPUT.PUT\_LINE (CHR(10));**

**FOR I IN (SELECT \* FROM ITEMS) LOOP**

**IF I.P\_CODE=ITEM\_CODE THEN**

**DBMS\_OUTPUT.PUT\_LINE ('P\_CODE: '||I.P\_CODE);**

**DBMS\_OUTPUT.PUT\_LINE ('DESCRIPT: '||I.DESCR);**

**DBMS\_OUTPUT.PUT\_LINE ('IN\_DATE: '||I.IN\_DATE);**

**DBMS\_OUTPUT.PUT\_LINE ('MIN\_QTY: '||I.MIN\_QTY);**

**DBMS\_OUTPUT.PUT\_LINE ('QTY: '||I.QTY);**

**DBMS\_OUTPUT.PUT\_LINE ('PRICE: '||I.PRICE);**

**DBMS\_OUTPUT.PUT\_LINE ('V\_CODE: '||I.V\_CODE);**

**DBMS\_OUTPUT.PUT\_LINE (CHR(10));**

**END IF;**

**END LOOP;**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE ('INVALID ITEM\_CODE');**

**WHEN NO\_ROW THEN**

**DBMS\_OUTPUT.PUT\_LINE ('ITEM IS NOT PRESENT ');**

**END;**

**/**

Procedure created.

**EXEC SHOW\_ITEM\_TMR\_HANDLED('HH15P');**

MULTIPLE RECORDS..................

P\_CODE: HH15P

DESCRIPT: NEW ITEM2

IN\_DATE: 30-SEP-20

MIN\_QTY: 2

QTY: 150

PRICE: 25

V\_CODE:

P\_CODE: HH15P

DESCRIPT: NEW ITEM..

IN\_DATE: 28-MAR-17

MIN\_QTY: 12

QTY: 100

PRICE: 5.8

V\_CODE:

PL/SQL procedure successfully completed.

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

**VIVA-VOCE**

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

**Question 1. State the advantages of using stored functions and procedures**

**Answer -** Stored procedures provide several advantages

1. To help you build powerful database applications

2. Better performance

3. Higher productivity

4. Ease of use

5. Increased scalability

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

**Question 2. Explain about IN,OUT and IN OUT variables in PL/SQL Procedures**

**Answer-**

**IN:** A variable passed in this mode is of read only nature. This is to say, the value cannot be changed and its scope is restricted within the procedure. The procedure receives a value from this argument when the procedure is called.

**OUT:** In this mode, a variable is write only and can be passed back to the calling program. It cannot be read inside the procedure and needs to be assigned a value.

**INOUT:** This procedure has features of both IN and OUT mode. The procedure can also read the variables value and can also change it to pass it to the calling function.

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

**Question 3.**  **Differentiate between a stored function and stored procedure.**

**Answer-**

* The function must return a value but in **Stored Procedure** it is optional. Even a procedure can return zero or n values.
* Functions can have only input parameters for it whereas Procedures can have input or output parameters.
* Functions can be called from Procedure whereas Procedures cannot be called from a Function.

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

**Question 4. Write about the RAISE\_APPLICATION\_ERROR() procedure of Oracle.**

**Answer-**

The procedure RAISE\_APPLICATION\_ERROR() allows you to issue an user-defined error from a code block or stored program.

By using this procedure, you can report errors to the callers instead of returning unhandled exceptions.

The RAISE\_APPLICATION\_ERROR() has the following syntax:

raise\_application\_error(

error\_number,

message

[, {TRUE | FALSE}]

);

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

**INFERENCES**

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

* We learnt to execute stored functions and procedures.
* We also learnt to handle exceptions and about to how declare user defined exceptions.

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