**\*\*\*\*\*\*\*\*\*\* EXPERIMENT: 06 \*\*\*\*\*\*\*\*\*\***

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

**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.

**Author : Mehul Y Khandhadiya**

**Roll no : 55**

**Date : 03-Oct-2020**

**Queries Set**

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

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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 IN EMPP.EID%TYPE,EMP\_DETAILS OUT VARCHAR2) AS**

**BEGIN**

**SELECT EID||' '||ENAME||' '||HIREDATE||' '||DESIGNATION||' '||SALARY**

**INTO EMP\_DETAILS**

**FROM EMPP**

**WHERE EID=ENO;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE DOES NOT EXIST....');**

**END;**

**/**

Procedure created.

**DECLARE**

**ENO NUMBER := &EID;**

**EMP\_DETAILS VARCHAR2(100);**

**BEGIN**

**SHOW\_EMPLOYEE(ENO,EMP\_DETAILS);**

**DBMS\_OUTPUT.PUT\_LINE(EMP\_DETAILS);**

**END;**

**/**

Enter value for eid: 7101

old 2: ENO NUMBER := &EID;

new 2: ENO NUMBER := 7101;

7101 Eugene Sabatini 10-OCT-06 Professor 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.**

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

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

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

**CREATE OR REPLACE PROCEDURE ADD\_EMPLOYEE**

**(ENO IN EMPP.EID%TYPE,E\_NAME IN EMPP.ENAME%TYPE,H\_DATE IN EMPP.HIREDATE%TYPE,DESG IN EMPP.DESIGNATION%TYPE,SLRY IN EMPP.SALARY%TYPE) AS**

**BEGIN**

**INSERT INTO EMPP**

**VALUES(ENO,E\_NAME,H\_DATE,DESG,SLRY);**

**DBMS\_OUTPUT.PUT\_LINE('DATA INSERTED SUCCESFULLY....');**

**END;**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('ADD\_EMPLOYEE');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

ADD\_EMPLOYEE 76630 PROCEDURE VALID

**DECLARE**

**ENO NUMBER := &EID;**

**E\_NAME EMPP.ENAME%TYPE:= &ENAME;**

**H\_DATE EMPP.HIREDATE%TYPE:= &HIREDATE;**

**DESG EMPP.DESIGNATION%TYPE:= &DESIGNATION;**

**SLRY EMPP.SALARY%TYPE:= &SALARY;**

**BEGIN**

**ADD\_EMPLOYEE(ENO,E\_NAME,H\_DATE,DESG,SLRY);**

**END;**

Enter value for eid: 7118

old 2: ENO NUMBER := &EID;

new 2: ENO NUMBER := 7118;

Enter value for ename: 'Mehul Khandhadiya'

old 3: E\_NAME EMPP.ENAME%TYPE:= &ENAME;

new 3: E\_NAME EMPP.ENAME%TYPE:= 'Mehul Khandhadiya';

Enter value for hiredate: '07-Jul-2020'

old 4: H\_DATE EMPP.HIREDATE%TYPE:= &HIREDATE;

new 4: H\_DATE EMPP.HIREDATE%TYPE:= '07-Jul-2020';

Enter value for designation: 'Teaching Asst.'

old 5: DESG EMPP.DESIGNATION%TYPE:= &DESIGNATION;

new 5: DESG EMPP.DESIGNATION%TYPE:= 'Teaching Asst.';

Enter value for salary: 25000

old 6: SLRY EMPP.SALARY%TYPE:= &SALARY;

new 6: SLRY EMPP.SALARY%TYPE:= 25000;

DATA INSERTED SUCCESFULLY....

PL/SQL procedure successfully completed.

Enter value for eid: 7119

old 2: ENO NUMBER := &EID;

new 2: ENO NUMBER := 7119;

Enter value for ename: 'Atulya Bharat'

old 3: E\_NAME EMPP.ENAME%TYPE:= &ENAME;

new 3: E\_NAME EMPP.ENAME%TYPE:= 'Atulya Bharat';

Enter value for hiredate: '03-Aug-2005'

old 4: H\_DATE EMPP.HIREDATE%TYPE:= &HIREDATE;

new 4: H\_DATE EMPP.HIREDATE%TYPE:= '03-Aug-2005';

Enter value for designation: 'Professor'

old 5: DESG EMPP.DESIGNATION%TYPE:= &DESIGNATION;

new 5: DESG EMPP.DESIGNATION%TYPE:= 'Professor';

Enter value for salary: 162000

old 6: SLRY EMPP.SALARY%TYPE:= &SALARY;

new 6: SLRY EMPP.SALARY%TYPE:= 162000;

DATA INSERTED SUCCESFULLY....

PL/SQL procedure successfully completed.

**SELECT \* FROM EMPP**

**WHERE EID IN (7118, 7119);**

EID ENAME HIREDATE DESIGNATION SALARY

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

7118 Mehul Khandhadiya 07-JUL-20 Teaching Asst. 25000

7119 Atulya Bharat 03-AUG-05 Professor 162000

2 rows selected.

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

**(E\_NAME IN EMPP.ENAME%TYPE) AS**

**ENO EMPP.EID%TYPE;**

**BEGIN**

**SELECT EID INTO ENO FROM EMPP**

**WHERE E\_NAME=LOWER(ENAME);**

**DELETE FROM EMPP**

**WHERE EID=ENO;**

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

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE DOES NOT EXIST....');**

**END;**

**/**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('REMOVE\_EMPLOYEE');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

REMOVE\_EMPLOYEE 76631 PROCEDURE VALID

**DECLARE**

**E\_NAME EMPP.ENAME%TYPE:= &ENAME;**

**BEGIN**

**REMOVE\_EMPLOYEE(E\_NAME);**

**END;**

**/**

Enter value for ename: 'mehul khandhadiya'

old 2: E\_NAME EMPP.ENAME%TYPE:= &ENAME;

new 2: E\_NAME EMPP.ENAME%TYPE:= 'mehul khandhadiya';

DATA DELETED SUCCESFULLY....

PL/SQL procedure successfully completed.

SQL> /

Enter value for ename: 'kurhikarji sir'

old 2: E\_NAME EMPP.ENAME%TYPE:= &ENAME;

new 2: E\_NAME EMPP.ENAME%TYPE:= ' kurhikarji sir';

EMPLOYEE DOES NOT EXIST....

PL/SQL procedure successfully completed.

**SELECT \* FROM EMPP;**

EID ENAME HIREDATE DESIGNATION SALARY

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

7101 Eugene Sabatini 10-OCT-06 Professor 150000

7102 Samantha Jones 08-NOV-06 Professor 146500

7103 Alexander Lloyd 01-FEB-07 Professor 148000

7104 Simon Downing 01-SEP-07 Professor 138400

7105 Christina Mulboro 15-JUL-08 Asso. Professor 127400

7106 Dolly Silverline 17-AUG-08 Asso. Professor 127400

7107 Christov Plutnik 01-SEP-08 Asso. Professor 127400

7108 Ellena Sanchez 12-NOV-09 Asso. Professor 119700

7109 Martina Jacobson 15-NOV-09 Asst. Professor 91000

7110 William Smithfield 23-JUN-10 Asst. Professor 86400

7111 Albert Greenfield 12-JUL-16 Research Asst. 48200

EID ENAME HIREDATE DESIGNATION SALARY

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

7112 James Washington 22-AUG-17 Research Asst. 44600

7113 Julia Martin 01-DEC-18 Teaching Asst. 35600

7114 Larry Gomes 18-MAY-19 Teaching Asst. 32850

7115 Svetlana Sanders 15-JAN-20 Teaching Asst. 30000

7116 Lovelyn Brendon 17-JUL-20 Teaching Asst. 30000

7117 Hector Hercules 01-AUG-20 Teaching Asst. 32200

7119 Atulya Bharat 03-AUG-05 Professor 162000

18 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.**

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

**CREATION OF ITEMS TABLE:-**

**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.

**SELECT \* FROM ITEMS;**

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

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

SH200 Sledge Hammer 05-JUL-20 3 10 25.8

ZZ999 Cordless Drill 10-JUL-20 40 200 25.5

AB212 Power Drill 03-AUG-20 3 15 275 24992

AB112 Power Drill 03-NOV-19 5 8 62.95 25595

SB725 7.25in Saw Blade 13-DEC-19 15 32 14.99 21344

SB900 9.00 in Saw Blade 13-NOV-19 12 18 17.49 21344

CL025 Hrd. Spring 1/4in 15-JAN-20 8 15 62.95 23119

CL050 Hrd. Spring 1/2in 15-JAN-20 5 23 62.95 23119

JB012 Jigsaw 12in Blade 30-DEC-19 5 8 109.92 24288

JB008 Jigsaw 8in Blade 24-DEC-19 5 10 123.45 24288

CD00X Cordless Drill 20-JAN-20 5 12 62.95 25595

CH10X Claw Hammer 20-JAN-20 10 23 9.95 21225

SH100 Sledge Hammer 02-JAN-20 5 8 14.4

RF100 Rat Tail File 15-DEC-19 20 43 4.99 21344

HC100 Hicut Chain Saw 07-FEB-20 5 11 256.99 24288

PP101 PVC Pipe 20-FEB-20 75 188 5.87

MC001 Metal Screw 01-MAR-20 75 172 6.99 21225

WC025 2.5in wide Screw 24-FEB-20 100 237 8.45 21231

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

HW15X HiVeld Hammer 10-JAN-20 15 60 15.5 24992

AB111 Power Drill 20-AUG-20 5 15 125 24992

PP102 PVC Pipe 21-AUG-20 12 50 15.25 24992

22 rows selected.

**ALTER TABLE ITEMS**

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

Table altered.

**ALTER TABLE ITEMS**

**MODIFY IN\_DATE DEFAULT SYSDATE;**

Table altered.

**ALTER TABLE ITEMS**

**MODIFY MIN\_QTY DEFAULT 2;**

Table altered.

**DESC ITEMS;**

Name Null? Type

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

P\_CODE NOT NULL CHAR(5)

DESCR NOT NULL VARCHAR2(30)

IN\_DATE NOT NULL DATE

MIN\_QTY NOT NULL NUMBER(3)

QTY NOT NULL NUMBER(4)

PRICE NOT NULL NUMBER(6,2)

V\_CODE NUMBER(5)

**ACTUAL QUERY:-**

**CREATE OR REPLACE FUNCTION CHECK\_ITEM**

**(PCODE IN ITEMS.P\_CODE%TYPE)**

**RETURN NUMBER AS**

**V\_KNT NUMBER;**

**BEGIN**

**SELECT COUNT(\*) INTO V\_KNT**

**FROM ITEMS WHERE P\_CODE=PCODE;**

**IF V\_KNT>0 THEN**

**RETURN 1;**

**ELSE**

**RETURN 0;**

**END IF;**

**END;**

**/**

Function created.

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM(&PCODE));**

**END;**

**/**

Enter value for pcode: 'PP102'

old 2: DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM(&PCODE));

new 2: DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM('PP102'));

1

PL/SQL procedure successfully completed.

SQL> /

Enter value for pcode: 'AB111'

old 2: DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM(&PCODE));

new 2: DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM('AB111'));

1

PL/SQL procedure successfully completed.

SQL> /

Enter value for pcode: 'AB222'

old 2: DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM(&PCODE));

new 2: DBMS\_OUTPUT.PUT\_LINE(CHECK\_ITEM('AB222'));

0

PL/SQL procedure successfully completed.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('CHECK\_ITEM');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

CHECK\_ITEM 76634 FUNCTION VALID

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 05 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Write a SQL code to compile and execute the stored procedure - ADDITEM, 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**

**(PCODE IN ITEMS.P\_CODE%TYPE,DESCP IN ITEMS.DESCR%TYPE,IN\_DT IN ITEMS.IN\_DATE%TYPE,**

**MN\_QTY IN ITEMS.MIN\_QTY%TYPE,QT IN ITEMS.QTY%TYPE,**

**PRC IN ITEMS.PRICE%TYPE,**

**VCODE IN ITEMS.V\_CODE%TYPE) AS**

**BEGIN**

**INSERT INTO ITEMS**

**VALUES(PCODE,DESCP,IN\_DT,MN\_QTY,QT,PRC,VCODE);**

**DBMS\_OUTPUT.PUT\_LINE('DATA INSERTED SUCCESFULLY....');**

**END;**

**/**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('ADD\_ITEM');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

ADD\_ITEM 76635 PROCEDURE VALID

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

Enter value for p\_code: 'AB001'

Enter value for descr: 'Some Tool'

Enter value for in\_date: '01-OCT-20'

Enter value for min\_qty: 1

Enter value for qty: 10

Enter value for price: 34.0

Enter value for v\_code: 21225

DATA INSERTED SUCCESFULLY....

PL/SQL procedure successfully completed.

**SELECT \* FROM ITEMS**

**WHERE P\_CODE = 'AB001';**

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

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

AB001 Some Tool 01-OCT-20 1 10 34 21225

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

**(PCODE IN ITEMS.P\_CODE%TYPE,QT IN ITEMS.QTY%TYPE,**

**PRC IN ITEMS.PRICE%TYPE) AS**

**BEGIN**

**IF CHECK\_ITEM(PCODE)=0 THEN**

**RAISE NO\_DATA\_FOUND;**

**ELSE**

**UPDATE ITEMS**

**SET QTY = QT,PRICE = PRC**

**WHERE ITEMS.P\_CODE=PCODE;**

**DBMS\_OUTPUT.PUT\_LINE('DATA UPDATED SUCCESFULLY....');**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('ITEM DOES NOT EXIST....');**

**END;**

**/**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('UPDATE\_ITEM');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

UPDATE\_ITEM 76636 PROCEDURE VALID

**BEGIN**

**UPDATE\_ITEM(&PCODE,&QTY,&PRICE);**

**END;**

Enter value for pcode: 'PP102'

Enter value for qty: 30

Enter value for price: 16.25

old 2: UPDATE\_ITEM(&PCODE,&QTY,&PRICE);

new 2: UPDATE\_ITEM('PP102',30,16.25);

DATA UPDATED SUCCESFULLY....

PL/SQL procedure successfully completed.

**SELECT \* FROM ITEMS**

**WHERE P\_CODE='PP102';**

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

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

PP102 PVC Pipe 16-AUG-20 12 30 16.25 24992

Enter value for pcode: 'AB222'

Enter value for qty: 10

Enter value for price: 10

old 2: UPDATE\_ITEM(&PCODE,&QTY,&PRICE);

new 2: UPDATE\_ITEM('AB222',10,10);

ITEM DOES NOT EXIST....

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**

**(PCODE IN ITEMS.P\_CODE%TYPE,QT IN ITEMS.QTY%TYPE,**

**PRC IN ITEMS.PRICE%TYPE) AS**

**BEGIN**

**IF CHECK\_ITEM(PCODE)=0 THEN**

**INSERT INTO ITEMS**

**VALUES(PCODE,'NEW ITEM...',SYSDATE,8,QT,PRC,NULL);**

**DBMS\_OUTPUT.PUT\_LINE('DATA INSERTED SUCCESFULLY....');**

**ELSE**

**UPDATE ITEMS**

**SET QTY = QT, PRICE = PRC**

**WHERE ITEMS.P\_CODE=PCODE;**

**DBMS\_OUTPUT.PUT\_LINE('DATA UPDATED SUCCESFULLY....');**

**END IF;**

**END;**

**/**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

UPDATE\_ITEM\_ADD 76637 PROCEDURE VALID

\_WHEN\_NOT\_FOUND

**BEGIN**

**UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND(&PCODE,&QTY,&PRICE);**

**END;**

Enter value for pcode: 'PP102'

Enter value for qty: 20

Enter value for price: 18.28

old 2: UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND(&PCODE,&QTY,&PRICE);

new 2: UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND('PP102',20,18.28);

DATA UPDATED SUCCESFULLY....

PL/SQL procedure successfully completed.

SQL> /

Enter value for pcode: 'AB222'

Enter value for qty: 20

Enter value for price: 30.30

old 2: UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND(&PCODE,&QTY,&PRICE);

new 2: UPDATE\_ITEM\_ADD\_WHEN\_NOT\_FOUND('AB222',20,30.30);

DATA INSERTED SUCCESFULLY....

PL/SQL procedure successfully completed.

**SELECT \* FROM ITEMS;**

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

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

PP102 PVC Pipe 16-AUG-20 12 50 15.25 24992

AB222 NEW ITEM... 01-OCT-20 8 20 30.3

23 rows selected.

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

**(PCODE IN ITEMS.P\_CODE%TYPE) AS**

**DETAILS VARCHAR2(100);**

**BEGIN**

**IF CHECK\_ITEM(PCODE)=0 THEN**

**RAISE NO\_DATA\_FOUND;**

**ELSE**

**SELECT P\_CODE||' '||DESCR||' '||IN\_DATE||' '||**

**MIN\_QTY||' '||QTY||' '||PRICE||' '||V\_CODE**

**INTO DETAILS FROM ITEMS**

**WHERE P\_CODE=PCODE;**

**DBMS\_OUTPUT.PUT\_LINE(DETAILS);**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('ITEM DOES NOT EXIST....');**

**END;**

**/**

Procedure created.

**SELECT OBJECT\_NAME, OBJECT\_ID, OBJECT\_TYPE, STATUS FROM USER\_OBJECTS**

**WHERE OBJECT\_NAME IN ('SHOW\_ITEM');**

OBJECT\_NAME OBJECT\_ID OBJECT\_TYPE STATUS

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

SHOW\_ITEMS 76638 PROCEDURE VALID

**BEGIN**

**SHOW\_ITEM(&PCODE);**

**END;**

Enter value for pcode: 'PP102'

old 2: SHOW\_ITEM(&PCODE);

new 2: SHOW\_ITEM('PP102');

PP102 PVC Pipe 16-AUG-20 12 50 15.25 24992

PL/SQL procedure successfully completed.

SQL> /

Enter value for pcode: 'AB222'

old 2: SHOW\_ITEM(&PCODE);

new 2: SHOW\_ITEM('AB222');

ITEM DOES NOT EXIST....

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.**

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

**PREREQUISITE**

**ALTER TABLE ITEMS**

**DROP PRIMARY KEY;**

Table altered.

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

DATA INSERTED SUCCESFULLY....

PL/SQL procedure successfully completed.

**ORIGINAL QUERY-**

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

**(PCODE IN ITEMS.P\_CODE%TYPE) AS**

**DETAILS VARCHAR2(100);**

**V\_KNT NUMBER;**

**BEGIN**

**SELECT COUNT(\*) INTO V\_KNT**

**FROM ITEMS WHERE P\_CODE=PCODE;**

**IF V\_KNT=1 THEN**

**SELECT P\_CODE||' '||DESCR||' '||IN\_DATE||' '||**

**MIN\_QTY||' '||QTY||' '||PRICE||' '||V\_CODE**

**INTO DETAILS FROM ITEMS**

**WHERE P\_CODE=PCODE;**

**DBMS\_OUTPUT.PUT\_LINE(DETAILS);**

**ELSE**

**RAISE TOO\_MANY\_ROWS;**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('ITEM DOES NOT EXIST....');**

**WHEN TOO\_MANY\_ROWS THEN**

**DBMS\_OUTPUT.PUT\_LINE(PCODE||' '||'MULTIPLE ROWS...........');**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('SOME ERROR OCCURED.....');**

**END;**

**/**

Procedure created.

**BEGIN**

**SHOW\_ITEM\_TMR\_E(&PCODE);**

**END;**

Enter value for pcode: 'HH15P'

old 2: SHOW\_ITEM\_TMR\_E(&PCODE);

new 2: SHOW\_ITEM\_TMR\_E('HH15P');

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

PL/SQL procedure successfully completed.

SQL> /

Enter value for pcode: 'HW15X'

old 2: SHOW\_ITEM\_TMR\_E(&PCODE);

new 2: SHOW\_ITEM\_TMR\_E('HW15X');

HW15X HiVeld Hammer 10-JAN-20 15 60 15.5 24992

PL/SQL procedure successfully completed.

**BEGIN**

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

**SHOW\_ITEM\_TMR\_E('HW15X');**

**END;**

**/**

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

HW15X HiVeld Hammer 10-JAN-20 15 60 15.5 24992

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* QUERY – 10 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Now extend the procedure in Query-09 as SHOW\_ITEM\_TMR\_HANDL ED to print the rows returned by the SELECT query after catching the appropriate exception.number from console. You should only report the violations.**

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

**CREATE OR REPLACE PROCEDURE SHOW\_ITEM\_TMR\_HANDELED**

**(PCODE IN ITEMS.P\_CODE%TYPE) AS**

**DETAILS VARCHAR2(100);**

**V\_KNT NUMBER;**

**BEGIN**

**SELECT COUNT(\*) INTO V\_KNT**

**FROM ITEMS WHERE P\_CODE=PCODE;**

**IF V\_KNT>1 THEN**

**SHOW\_ITEM\_TMR\_E(PCODE);**

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

**IF(T.P\_CODE=PCODE) THEN**

**DBMS\_OUTPUT.PUT\_LINE(t.P\_CODE||' '||t.DESCR||'**

**'||t.IN\_DATE||' '||t.MIN\_QTY||' '||t.QTY||'**

**'||t.PRICE||' '||t.V\_CODE);**

**END IF;**

**END LOOP;**

**ELSE**

**SELECT P\_CODE||' '||DESCR||' '||IN\_DATE||' '||**

**MIN\_QTY||' '||QTY||' '||PRICE||' '||V\_CODE**

**INTO DETAILS FROM ITEMS WHERE PCODE=P\_CODE;**

**DBMS\_OUTPUT.PUT\_LINE(DETAILS);**

**END IF;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('ITEM DOES NOT EXIST....');**

**WHEN TOO\_MANY\_ROWS THEN**

**DBMS\_OUTPUT.PUT\_LINE(PCODE||' '||'MULTIPLE ROWS...........');**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('SOME ERROR OCCURED.....');**

**END;**

**/**

**BEGIN**

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

**SHOW\_ITEM\_TMR\_HANDELED('HW15X');**

**END;**

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

HH15P NEW ITEM-2 01-OCT-20 2 150 25

HH15P NEW ITEM 01-OCT-20 2 178 20

HW15X HiVeld Hammer 10-JAN-20 15 60 15.5 24992

PL/SQL procedure successfully completed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VIVA-VOCE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Q1 - State the advantages of using stored functions and**

**procedures.**

To help you build powerful database applications, stored procedures provide several advantages including better performance, higher productivity, ease of use, and increased scalability.

**Q2 - Explain about IN,OUT and IN OUT variables in PL/SQL**

**procedures.**

**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.

**Q3 - Differentiate between a stored function and stored**

**procedures.**

* 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.

**Q4 - Write about the RAISE\_APPLICAATION\_ERROR() procedure of**

**oracle.**

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 exception.

**INFERENCE:** We learnt about creation of anonymous blocks and procedures using

PL/SQL and implemented the queries regarding it. We also learnt about

exception handling within the blocks. Outputs were recorded and

analysed.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***