*** EXPERIMENT NO: 06 ***

Author : Pradyot Patil

Roll No : 53 [6B]

Date : 5-April-2018

****

QUERY-01: Write the SQL code to insert the following employee – 7010, Svetlana, Sanders, 15-OCT-1984, M, 44MA123456, 15JAN-2006, 8000, Machine Vision, Lecturer What are the problems encountered? Fix and reinsert the record. Save the EMPLOYEE table and check if the updates are saved.

/*

Ensure that you are logged in as a user "CS6xx" and not as SYSTEM or SYS or SYSDBA user. Create table named TEST_TBL with attributes REC_NO, CURR_DT representing record identifier - an integer ranging between 101 thru 999, and current date for the record - to be SYSDATE. Enforce entity integrity on REC_NO. Test for creation of table and various constraints on it. Before you execute any PL/SQL block, you must enable the PL/SQL output using the command: SET SERVEROUTPUT ON

*/

****

```
INSERT INTO EMPLOYEE VALUES(7010, 'Svetlana', 'Sanders' ,'15-OCT-1984', 'M',
'44MA123456', '15-JAN-2006', 8000.00, 'Machine Vision','Lecturer') ;

ERROR at line 1:
ORA-02290: check constraint (CS653.EMPLOYEE_CK_SALARY) violated

INSERT INTO EMPLOYEE VALUES(7010, 'Svetlana', 'Sanders' ,'15-OCT-1984', 'M',
'44MA123456', '15-JAN-2006', 10000.00, 'Machine Vision','Lecturer') ;

ERROR at line 1:
ORA-02290: check constraint (CS653.EMPLOYEE_CK_EID) violated

INSERT INTO EMPLOYEE VALUES(7110, 'Svetlana', 'Sanders' ,'15-OCT-1984', 'M',
'44MA123456', '15-JAN-2006', 10000.00, 'Machine Vision','Lecturer') ;
1 row created.

SELECT EID,FNAME,LNAME<SALARY FROM EMPLOYEE WHERE EID=7110 ;
EID             FNAME      LNAME          SALARY

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

      7110 Svetlana   Sanders         10000

1 row selected.
```

```
**************************************************************************
QUERY-02 Write a SQL code to write and execute an anonymous PL/SQL block that will
insert 10 tuples into TEST_TBL. Ensure to commit the populated records. Test the
insertion in TEST_TBL by displaying its contents.  /* Create a table EMPP (contains
no records at creation) that includes EID, ENAME (column combining FNAME and LNAME
with embedded blank), HIREDATE and SALARY from EMPLOYEE table. Enforce entity
integrity constraints on EID. Verify table creation, contents and constraints.
*/**************************************************************************
CREATE TABLE TEST_TBL(
       REC_NO NUMBER(3),
       CURR_DT DATE,
       CONSTRAINT TEST_TBL_PK_REC_NO PRIMARY KEY(REC_NO),
       CONSTRAINT TEST_TBL_CK_REC_NO CHECK(REC_NO >100 AND REC_NO<1000)
       );
Table created.


DECLARE
       V_CODE NUMBER;
       V_DATE DATE;
BEGIN
       FOR V_CODE IN 1..10 LOOP
              SELECT SYSDATE INTO V_DATE FROM DUAL;
              INSERT INTO TEST_TBL VALUES (V_CODE+100,V_DATE);
       END LOOP;
END;
/

PL/SQL procedure successfully completed.

select * from test_tbl;
    REC_NO   CURR_DT
---------- ---------
       101 05-APR-18
       102 05-APR-18
       103 05-APR-18
       104 05-APR-18
       105 05-APR-18
       106 05-APR-18
       107 05-APR-18
       108 05-APR-18
       109 05-APR-18
       110 05-APR-18
```

10 rows selected.

```
CREATE TABLE EMPP AS
       (SELECT EID,FNAME||' '||LNAME AS ENAME,HIREDATE,SALARY
       FROM EMPLOYEE
       WHERE 1=2
       );
```

Table created.

```
ALTER TABLE EMPP
ADD CONSTRAINTS EMPP_PK_EID PRIMARY KEY(EID);
```

Table altered.

```
DESC EMPP
```

Name	Null?	Type
EID	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(21)
HIREDATE	NOT NULL	DATE
SALARY	NOT NULL	NUMBER(7,2)

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

COUNT(*)
0

1 row selected.

```
**************************************************************************
QUERY-03: Write a SQL code to write and execute an anonymous PL/SQL block that will
use %TYPE variables to populate the EMPP table with corresponding tuples in
EMPLOYEE table. /* Remove all records from EMPP table before executing Query-03. */
**************************************************************************
DECLARE
       V_EID EMPLOYEE.EID%TYPE;
       V_ENAME EMPP.ENAME%TYPE;
       V_HIREDATE EMPLOYEE.HIREDATE%TYPE;
       V_SALARY EMPLOYEE.SALARY%TYPE;
       V_CODE NUMBER;
```

```
BEGIN
       V_EID:=7100;
       FOR V_CODE IN 1..10 LOOP
             V_EID:=V_EID+V_CODE;
             SELECT(FNAME||''||LNAME),HIREDATE,SALARY INTO
             V_ENAME,V_HIREDATE,V_SALARY FROM EMPLOYEE WHERE EMPLOYEE.EID=V_EID;
             INSERT INTO EMPP VALUES (V_EID,V_ENAME,V_HIREDATE,V_SALARY);
       END LOOP;
END;
/

PL/SQL procedure successfully completed.
SELECT COUNT(*) FROM EMPP;
  COUNT(*)
----------
        10
1 row selected.

DELETE FROM EMPP WHERE 1=1;

10 rows deleted.
********************************************************************************
QUERY-04: Write a SQL code to write and execute an anonymous PL/SQL block that will
use %ROWTYPE variables to populate the EMPP table with corresponding tuples in
EMPLOYEE table.
********************************************************************************
DECLARE
       V_EMPP EMPLOYEE%ROWTYPE;
       V_CODE NUMBER;
BEGIN
       FOR V_CODE IN 1..10 LOOP
             SELECT * INTO V_EMPP FROM EMPLOYEE WHERE EMPLOYEE.EID = V_CODE+7100;
             INSERT INTO EMPP VALUES (V_EMPP.EID,V_EMPP.FNAME||'
'||V_EMPP.LNAME,V_EMPP.HIREDATE,V_EMPP.SALARY);
       END LOOP;
END;
/
PL/SQL procedure successfully completed.
SELECT COUNT(*) FROM EMPP;
  COUNT(*)
----------
        10
1 row selected.
```

****

QUERY-05: Write a SQL code to write and execute an anonymous PL/SQL block that will locate the first November-born employee

****

```
DECLARE
       V_CODE NUMBER ;
       V_DATE VARCHAR(20) ;
BEGIN
       FOR V_CODE IN 1..10 LOOP
              SELECT TO_CHAR(TO_DATE(BIRTHDATE,'DD--MM-YY'),'MONTH')
                     INTO V_DATE FROM EMPLOYEE WHERE EID=7100+V_CODE ;
              IF V_DATE LIKE 'NOVEMBER%' THEN
                     DBMS_OUTPUT.PUT_LINE('EID: '|| TO_CHAR(7100+V_CODE)) ;
                     EXIT ;
              END IF ;
       END LOOP ;
END ;
/
```

EID: 7106

****

QUERY-06: Write a SQL code to write and execute an anonymous PL/SQL block that will locate the first November-born employee, when EMPLOYEE table is searched in reversed order.

****

```
DECLARE
       V_CODE NUMBER ;
       V_DATE VARCHAR(20) ;
BEGIN
       FOR V_CODE IN 1..10 LOOP
              SELECT TO_CHAR(TO_DATE(BIRTHDATE,'DD--MM-YY'),'MONTH')
                     INTO V_DATE FROM EMPLOYEE WHERE EID=7111-V_CODE ;
              IF V_DATE LIKE 'NOVEMBER%' THEN
                     DBMS_OUTPUT.PUT_LINE('EID: '|| TO_CHAR(7111-V_CODE)) ;
                     EXIT ;
              END IF ;
       END LOOP ;
END ;
/
```

EID: 7107

****

QUERY-07: Write a SQL code to write and execute an anonymous PL/SQL block that accept an employee number from the console and will display employee information for said employee (minimal output -- Employee Number, Name of Employee, Designation, Salary). A system exception, NO_DATA_FOUND should be cached when the mentioned employee does not exist.

****

```
DECLARE
	V_EID EMPLOYEE.EID%TYPE;
	V_ENAME EMPLOYEE.FNAME%TYPE;
	V_DESIGNATION EMPLOYEE.DESIGNATION%TYPE;
	V_SALARY EMPLOYEE.SALARY%TYPE;
BEGIN
	SELECT EID,FNAME,DESIGNATION,SALARY INTO
V_EID,V_ENAME,V_DESIGNATION,V_SALARY
		FROM EMPLOYEE
		WHERE EID=&EIDCODE ;
		DBMS_OUTPUT.PUT_LINE(V_EID||' , '||V_ENAME||' , '||V_DESIGNATION||' ,
		'||V_SALARY) ;
EXCEPTION
		WHEN NO_DATA_FOUND THEN
		DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
END ;
/
nter value for eidcode: 7106
old  11:		WHERE EID=&EIDCODE ;
new  11:		WHERE EID=7106 ;
7106 , William , Asst. Professor , 15660

PL/SQL procedure successfully completed.

Enter value for eidcode: 7111
old  11:		WHERE EID=&EIDCODE ;
new  11:		WHERE EID=7111 ;
NO DATA FOUND

PL/SQL procedure successfully completed.
```

****

QUERY-08: Write a SQL code to write and execute an anonymous PL/SQL block that defines user-defined exceptions - BELOW_PAY_RANGE and ABOVE_PAY_RANGE. Your script should accept an employee number from the console and check for the salary to fall within the payscale [minpay, maxpay]. If the salary is less than minpay, BELOW_PAY_RANGE exception is raised and when cached an appropriate message – '<EmpNo> Receives Salary Below Scale [minpay, maxpay]' is displayed; otherwise ABOVE_PAY_RANGE exception is raised and cached to display the appropriate message accordingly. You must appropriately catch the NO_DATA_FOUND exception also. When there are no violations, display for the employee the salary drawn. Test the above anonymous block for input employee numbers - 7101, 7105, 7109, 7110, 7111 and 7115.

****

```
DECLARE
	V_EMP NUMBER :=(&EMPLOYEE_NUM);
	CURR EMPLOYEE%ROWTYPE;
	BELOW_PAY_RANGE EXCEPTION;
	ABOVE_PAY_RANGE EXCEPTION;
	MINPAY NUMBER;
	MAXPAY NUMBER;
BEGIN
	MINPAY:=13000;
	MAXPAY:=16000;
	SELECT * INTO CURR FROM EMPLOYEE WHERE EMPLOYEE.EID = V_EMP;
	IF CURR.SALARY < MINPAY THEN
		RAISE BELOW_PAY_RANGE;
	ELSIF CURR.SALARY > MAXPAY THEN
		RAISE ABOVE_PAY_RANGE;
	ELSE
		DBMS_OUTPUT.PUT_LINE(CURR.EID||' '||CURR.FNAME||' '||CURR.LNAME||'
		'||CURR.SALARY);
	END IF;
EXCEPTION
	WHEN NO_DATA_FOUND THEN
		DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
	WHEN BELOW_PAY_RANGE THEN
		DBMS_OUTPUT.PUT_LINE(CURR.EID||' RECEIVES SALARY BELOW SCALE
		['||MINPAY||','||MAXPAY||']');
	WHEN ABOVE_PAY_RANGE THEN
		DBMS_OUTPUT.PUT_LINE(CURR.EID||' RECEIVES SALARY ABOVE SCALE
		['||MINPAY||','||MAXPAY||']');

END;
/
```

```
Enter value for employee_num: 7110
```

```
old   2:     V_EMP NUMBER :=(&EMPLOYEE_NUM);
new   2:     V_EMP NUMBER :=(7110);
7110 RECEIVES SALARY BELOW SCALE [13000,16000]

PL/SQL procedure successfully completed.

Enter value for employee_num: 7111
old   2:     V_EMP NUMBER :=(&EMPLOYEE_NUM);
new   2:     V_EMP NUMBER :=(7111);
NO DATA FOUND

Enter value for employee_num: 7105
old   2:     V_EMP NUMBER :=(&EMPLOYEE_NUM);
new   2:     V_EMP NUMBER :=(7105);
7105RECEIVES SALARY ABOVE SCALE [13000,16000]
PL/SQL procedure successfully completed.
```

****

QUERY-09: Write a SQL code to write and execute an anonymous PL/SQL block that will modify Query-8** to process all records of EMPLOYEE table. You need not acquire employee number from console. You should only report the violations

****

```
DECLARE
       V_EMP NUMBER ;
       CURR EMPLOYEE%ROWTYPE;
       V_CODE NUMBER ;
       V_NUM NUMBER ;
       BELOW_PAY_RANGE EXCEPTION;
       ABOVE_PAY_RANGE EXCEPTION;
       MINPAY NUMBER;
       MAXPAY NUMBER;
BEGIN
       SELECT COUNT(*) INTO V_NUM FROM EMPLOYEE ;
       FOR V_CODE IN 1..V_NUM LOOP
       BEGIN
              MINPAY:=13000;
              MAXPAY:=16000;
              SELECT * INTO CURR FROM EMPLOYEE WHERE EID=7100+V_CODE ;
              IF CURR.SALARY < MINPAY THEN
                     RAISE BELOW_PAY_RANGE;
              ELSIF CURR.SALARY > MAXPAY THEN
                     RAISE ABOVE_PAY_RANGE;
              ELSE
```

```
                DBMS_OUTPUT.PUT_LINE(CURR.EID||''||CURR.FNAME||'
                '||CURR.LNAME||' '||CURR.SALARY);
          END IF;
    EXCEPTION
          WHEN BELOW_PAY_RANGE THEN
                DBMS_OUTPUT.PUT_LINE(CURR.EID||'  RECEIVES  SALARY  BELOW  SCALE
                ['||MINPAY||','||MAXPAY||']');
          WHEN ABOVE_PAY_RANGE THEN
                DBMS_OUTPUT.PUT_LINE(CURR.EID||'  RECEIVES  SALARY  ABOVE  SCALE
                ['||MINPAY||','||MAXPAY||']');
          WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('NO DATA FOUND');
    END;
    END LOOP;

END;
101 RECEIVES SALARY ABOVE SCALE [13000,16000]
7102AlbertGreenfield 14200
7103JuliaMartin 13320
7104MartinaJacobson 15550
7105 RECEIVES SALARY ABOVE SCALE [13000,16000]
7106WilliamSmithfield 15660
7107 RECEIVES SALARY ABOVE SCALE [13000,16000]
7108JamesWashington 14000
7109LarryGomes 13650
7110 RECEIVES SALARY BELOW SCALE  [13000,16000]

PL/SQL procedure successfully completed.
```