Date .....

Name Nikhil Gupta
Roll No. 20051523
Batch CSE-06
Branch CSE
Subject DBMS

Assignment 10

Ques 3 Create a PL/SQL block to increase salary of employees in department 10. The salary increase is 15% for employees making less than $100,000 & 10% for the employees making $100,000 or more. Use a cursor with a for update clause. Update salary with a where Current of clause in a cursor FOR Loop.

```
DECLARE
        incr_sal number;
        CURSOR myCur IS select employeeID, fname, salary
          from employee where deptID = 10
        FOR Update;

BEGIN
      FOR employee_rec IN myCur
        LOOP
          IF employee_rec.salary < 100000
            THEN
                incr_sal := .15;

            ELSE
                incr_sal := .10;

          ENDIF;
```

Spiral

Date .....

```
        UPDATE Employee set salary = salary + salary *
                                              incr_sal.
        Where CURRENT OF mycur;

    END LOOP;

END;
```

```
PL/SQL procedure successfully completed.

SQL> select fname, lname, hiredate,salary from employee;

FNAME                LNAME                HIREDATE      SALARY
-------------------- -------------------- --------- ----------
John                 Smith                15-APR-60     320650
Larry                Houstan              19-MAY-67     150000
Nikhil               Gupta                12-FEB-91     114066
Derek                Dave                 15-MAR-95      80000
Alex                 McCall               05-MAY-97      66500
```

Date .....

Ques 4 Pre-defined Server Exception

```
DECLARE
    v_emp employee%ROWTYPE;
    v_qual qualification%ROWTYPE;
    qualID number;

BEGIN
    select employee.fname, employee.qualID, qualification.qualDesc
    INTO v_emp.fname, v_emp.qualID, v_qual.qualdesc from
    employee INNER JOIN Qualification ON Employee.qualID =
    qualification.qualID where employee.qualID = &qualID;

    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_emp.fname);

    DBMS_OUTPUT.PUT_LINE('Qualification ID: ' || v_emp.qualID);

    DBMS_OUTPUT.PUT_LINE('Qualification: ' || v_qual.qualdesc);

EXCEPTION
    WHEN Too_Many_Rows THEN

    DBMS_OUTPUT.PUT_LINE('More than one employee with
                          such qualification');

    WHEN NO_DATA_FOUND THEN

    DBMS_OUTPUT.PUT_LINE('No Employee with such
                          qualification');

END;
```

```
Enter value for qualid: 4
old   8:     from EMPLOYEE INNER JOIN Qualification ON Employee.qualid
new   8:     from EMPLOYEE INNER JOIN Qualification ON Employee.qualid
Employee Name: Alex
Qualification ID: 4
Qualification: Associates

PL/SQL procedure successfully completed.
```

Date .....

Ques 5 User-defined Exception

```
DECLARE
    no_course EXCEPTION;
    CURSOR course_cur IS select * from course;
    crs_id course.CourseID%TYPE := '&CourseID';
    found Boolean := false;
    course_rec course%ROWTYPE;

BEGIN
    OPEN course_cur;
    FETCH course_cur INTO course_rec;

    WHILE course_cur%FOUND LOOP

        IF course_rec.courseID = crs_ID THEN
            found := TRUE;

        DBMS_OUTPUT.PUT_LINE('CourseID : ' || course_rec.
                                              courseID);

        DBMS_OUTPUT.PUT_LINE('Title ' || course_rec.
                                         ~~course~~title);

        DBMS_OUTPUT.PUT_LINE('Credits ' || course_rec.credits);

        DBMS_OUTPUT.PUT_LINE('PreReq' || course_rec.prereq);

    ENDIF;

    FETCH course_cur INTO course_rec;
```

Spiral

Date .....

```
    END LOOP;
  IF FOUND = False THEN
      Raise no_Course;
  END IF;
  CLOSE course_cur;
EXCEPTION
    WHEN no_course THEN
        DBMS_OUTPUT.PUT_LINE('No such course :' || crsid);
END;
```

```
Enter value for courseid: DB2002
old   5:      crs_id course.CourseId%TYPE := '&CourseId';
new   5:      crs_id course.CourseId%TYPE := 'DB2002';
Course Id: DB2002
Title      DBMS
Credits:   3
PreReq:    SQL

PL/SQL procedure successfully completed.
```

Date .....

Assignment 11

Ques 1 Procedure that takes student id as parameter & returns student fullname & phone number

Create Procedure

```
CREATE OR REPLACE PROCEDURE get_student
(in_studentID IN number, v_lname OUT varchar2,
 v_fname OUT varchar2, v_phone OUT varchar2)

IS
BEGIN
    select fname, lname, phone INTO v_fname, v_lname,
    v_phone from STUDENT where studentID =
                                    in_studentID;

EXCEPTION

  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('StudentID' || to_char
    (in_studentID) || 'does not exist');

END get_student;
```

Date .....

Access Procedure

```
DECLARE
    v_id student.StudentID%TYPE := '&Student_ID';

    v_lname student.lname%TYPE;

    v_fname student.fname%TYPE;

    v_phone student.phone%TYPE;

BEGIN

    get_student(v_id, v_lname, v_fname, v_phone);

    IF v_fname IS NOT NULL THEN

        DBMS_OUTPUT.PUTLINE('Name: '||v_fname||' '||
                                v_lname);

        DBMS_OUTPUT.PUTLINE('Phone: '||v_phone);

    END IF;

END;
```

```
Enter value for student_id: 00104
old   2:      v_id student.StudentId%TYPE := '&Student_Id';
new   2:      v_id student.StudentId%TYPE := '00104';
Name:  Brian Lee
Phone: 7623451235

PL/SQL procedure successfully completed.
```

Date .....

Ques 2 Write a function & pass a department number to it. If dept table does not contain that department number, return false value, otherwise return true.

Create Function

```
CREATE OR REPLACE FUNCTION find_dept(in_id IN
                                     NUMBER)

  Return Boolean

  IS
    v_deptname dept.DeptName %TYPE;

BEGIN
    select deptName INTO v_deptname FROM dept
     WHERE DeptID = in_id;

    IF v_deptname IS NOT NULL THEN
        Return true;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        Return False;
END;
```

Date .....

Access Function

```
DECLARE
    v_id dept.DeptID%TYPE := '&Department_ID';
    found BOOLEAN;

BEGIN
    found := find_dept(v_id);

    IF found = TRUE THEN

        DBMS_OUTPUT.PUT_LINE('Department' || v_id ||
                'exists');

    ELSE
        DBMS_OUTPUT.PUT_LINE('Department' || v_id || 'does not
                exist');

    END IF;

END;
```

```
Enter value for department_id: 10
old   2:        v_id dept.DeptId%TYPE := '&Department_Id';
new   2:        v_id dept.DeptId%TYPE := '10';
Department 10 exists
```

```
Enter value for department_id: 15
old   2:        v_id dept.DeptId%TYPE := '&Department_Id';
new   2:        v_id dept.DeptId%TYPE := '15';
Department 15 does not exist
```

Date .....

Ques 3 Create Package which contains procedure & function. The procedure will take room number as parameter & get capacity & building from Location. take care of exceptions.
Function will take csid as parameter & returns max no. of seats available in course section.

Package Specification

```
CREATE OR REPLACE PACKAGE college-package AS

  PROCEDURE find_room(i_roomno IN Location.RoomNo%
    TYPE, o_capacity OUT Location.Capacity%TYPE,
    o_bldg OUT Location.Building%TYPE);

  FUNCTION section_count(i_csid IN crssection.csid%TYPE)
      RETURN number;

END college-package;
```

Package Body

```
CREATE OR REPLACE PACKAGE BODY college_package
AS
PROCEDURE find_room(i_roomNo IN Location.RoomNo%TYPE,
o_capacity OUT Location.Capacity%TYPE, o_bldg OUT
Location.Building%TYPE) IS

BEGIN
  SELECT Capacity,Building INTO o_capacity, o_bldg
```

Spiral

Date .....

```
    FROM Location WHERE RoomNo = i_roomNo;

EXCEPTION

    WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE(i_roomNo || ' does not exist');

END find_room;


FUNCTION section_count(i_csid IN crssection.CsID%TYPE)
  Return NUMBER IS v_count number;

BEGIN
    SELECT MaxCount INTO v_count FROM crssection
    WHERE CsID = i_csid;
    Return v_count;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    Return -1;

END section_count;

END college_package;
```

Spiral

Date .....

Accessing Procedure in Package

```
DECLARE
    v_roomNo location.RoomNo%TYPE := &RoomNo;
    v_capacity location.Capacity%TYPE;
    v_bldg location.Building%TYPE;
BEGIN
    college_package.find_room(v_roomNo, v_capacity, v_bldg);
    IF v_capacity IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('RoomNo. : ' || v_roomNo);
        DBMS_OUTPUT.PUT_LINE('Capacity : ' || v_capacity);
        DBMS_OUTPUT.PUT_LINE('Building : ' || v_bldg);
    END IF;
END;
```

```
Enter value for roomno: 501
old   2: v_roomNo location.RoomNo%TYPE := &RoomNo;
new   2: v_roomNo location.RoomNo%TYPE := 501;
Room No.: 501
Capacity: 4
Building: WingA
```

Date .....

Accessing Function in Package

```
DECLARE
    v_csID crssection.CsID%TYPE := '&CsID';
    v_count number;

BEGIN
    v_count := college_package.section_count(v_csid);

    DBMS_OUTPUT.PUT_LINE('Maximum No. of seats in' ||
        v_csid ||' is: ' || v_count);

END;
```

```
Enter value for csid: 1101
old   2:      v_csid crssection.CsId%TYPE := '&CsID';
new   2:      v_csid crssection.CsId%TYPE := '1101';
Maximum No. of seats in 1101 is: 30
```