Name :- Singh Chandani Harendra

Class :- MCA (sem-1)

Division :- B

Roll No. :- 124

Paper No. :- 101

Subject :- RDBMS

Assignment No. :- 03

<u>Write PL/SQL code for the following :-</u>

1. Write a stored procedure that accepts deptno as an argument and then displays the report in the following format for this deptno.
Deptno: 99 Department Name: XXXXXXXXXX Location: XXXXXXXXXX
Empno Name Designation Salary

9999 XXXXXXX XXXXXXXX 999999

Total Salary: 999999
Highest Salary: 999999
Lowest Salary: 999999

```
CREATE OR REPLACE PROCEDURE show_dept_detail
( vdeptno IN dept.deptno%TYPE)
IS
BEGIN
        FOR i IN (select * from dept where deptno = vdeptno)
        LOOP
                dbms_output.put_line(' Department No. : ' || i.deptno ||' Department Name : '|| i.dname ||'
                Location : '|| i.loc);
                dbms_output.put_line('---------------------------------------------------');
                dbms_output.put_line('Empno       Name       Designation       Salary');
                dbms_output.put_line('---------------------------------------------------');
        END LOOP;
        FOR j IN (select * from emp where deptno = vdeptno order by empno)
        LOOP
                dbms_output.put_line(j.empno ||'        '||j.ename||'       '||j.job||'       '||j.sal);
        END LOOP;
        dbms_output.put_line('---------------------------------------------------');
        FOR k IN (select sum(sal) tot, max(sal) highsal, min(sal) lowsal from emp where deptno = vdeptno)
        LOOP
                dbms_output.put_line('                  Total Salary :       '||k.tot);
                dbms_output.put_line('                  Total Salary :       '||k.highsal);
                dbms_output.put_line('                  Total Salary :       '||k.lowsal);
        END LOOP;
        dbms_output.put_line('---------------------------------------------------');
END;
/
```

→ calling program

```
DECLARE
        vdeptno dept.deptno%type := :enter_deptno;
BEGIN
        show_dept_detail(vdeptno);
END;
```

2. Write a stored function that accepts empno as an argument & checks if the empno exists or not and return appropriate Boolean value.

```
CREATE OR REPLACE FUNCTION emp_exists
( vempno IN emp.empno%TYPE)
RETURN boolean
IS
        x number;
BEGIN
        select count(*) into x from emp where empno = vempno;

        IF x=1 THEN
                return TRUE;
        ELSE
                return FALSE;
        END IF;
END;
/
```

→ calling program

```
DECLARE
        vempno emp.empno%TYPE := :enter_empno;
        ans boolean;
BEGIN
        ans := emp_exists(vempno);
        IF ans = TRUE THEN
                dbms_output.put_line('Employee Exists..');
        ELSE
                dbms_output.put_line('Employee Not Exists...');
        END IF;
END;
```

3. Create a table Letter(deptno number(2), dept_name varchar2(20), last_letter_no number(4)). Write a function Gen_Letter_no(deptno, year_flag, subject_ref) which will generate letter number as per the following examples: • if year_flag is 'Y' then depending on the current system date take the current financial year

• subject_ref: the first character will be either A or B which would mean that the subject name should come After the year or Before the year (if year exists, i.e. if year_flag is 'Y') in the letter number, if year_flag is not 'Y' then the first character will have no effect

• finally, the last number is the last_letter_no+1 for the corresponding deptno fetched from Letter table and this new letter number (the numeric part only) must be updated in Letter table.

Gen_Letter_no(10, 'Y', NULL) COMP/11-12/1202
Gen_Letter_no(20, 'Y', 'ASALES') ACCOUNTS/11-12/SALES/1001
Gen_Letter_no(20, 'Y', 'BSALES') ACCOUNTS/SALES/11-12/1002
Gen_Letter_no(10, NULL, NULL) COMP/1203
Gen_Letter_no(10, NULL, 'ATENDER') COMP/TENDER/1204
Gen_Letter_no(10, NULL, 'BTENDER') COMP/TENDER/1205
i.e. for deptno argument take the corresponding department name from Letter table (assuming that the deptno exist in the table), and

→ Create table Letter

```
create table Letter(
   deptno number(2) primary key,
   dept_name varchar2(20),
   last_letter_no number(4)
);
```

→ Insert Data in table Letter

```
insert into Letter values(10,'COMP',1000);
insert into Letter values(20,'ACCOUNTS',1000);
```

→ Function

```
create or replace FUNCTION Gen_Letter_no
(vdeptno Letter.deptno%TYPE, year_flag char, subject_ref varchar2)
RETURN varchar2
IS
        drec Letter%ROWTYPE;
        ans varchar2(100);
        ref char;
        new_no number(4);
        fin_year varchar(10);
BEGIN
        select * into drec from Letter where deptno = vdeptno;
        ans := drec.dept_name;
        IF year_flag = 'Y' THEN
                ref := to_char(SUBSTR(subject_ref,1,1));
                IF TO_CHAR(sysdate,'MM') > '03' THEN
                        fin_year := TO_CHAR(sysdate,'YY')||'-'|| TO_CHAR(TO_NUMBER
                        (TO_CHAR(sysdate,'YY'))+1);
                ELSE
```

```
                    fin_year := TO_CHAR(TO_NUMBER(TO_CHAR(sysdate,'YY'))-1) ||'-'||
                    TO_CHAR(sysdate,'YY');
                END IF;
                IF subject_ref IS NULL THEN
                    ans := ans || '/' || fin_year;
                ELSE
                    IF ref = 'A' THEN
                        ans := ans || '/' || fin_year || '/' || SUBSTR(subject_ref,2);
                    ELSE
                        ans := ans || '/' ||SUBSTR(subject_ref,2)||' / '||fin_year ;
                    END IF;
                END IF;
        ELSE
                IF subject_ref IS NOT NULL THEN
                    ans := ans || '/' ||SUBSTR(subject_ref,2);
                END IF;
        END IF;
        new_no := drec.last_letter_no + 1;
        ans := ans || '/' ||new_no;
        update Letter set last_letter_no = last_letter_no+1 where deptno = vdeptno;
        return ans;
END;
/
```

→ Calling Program

```
begin
        dbms_output.put_line(Gen_Letter_no(10, 'Y', NULL));
        dbms_output.put_line(Gen_Letter_no(20, 'Y', 'ASALES'));
        dbms_output.put_line(Gen_Letter_no(20, 'Y', 'BSALES'));
        dbms_output.put_line(Gen_Letter_no(10, NULL, NULL));
        dbms_output.put_line(Gen_Letter_no(10, NULL, 'ATENDER'));
        dbms_output.put_line(Gen_Letter_no(10, NULL, 'BTENDER'));
end;
```

4. Write a package that will include the procedure & functions written in ques. 1, 2 and 3.

→ Package Specification

```
CREATE OR REPLACE PACKAGE dept_emp_pkg
IS

        PROCEDURE show_dept_detail
        (vdeptno IN dept.deptno%TYPE);

        FUNCTION emp_exists
        (vempno IN emp.empno%TYPE)
        RETURN boolean;

        FUNCTION Gen_Letter_no
        (vdeptno Letter.deptno%TYPE, year_flag char, subject_ref varchar2)
        RETURN varchar2;

END;
```

→ Package Body

```
CREATE OR REPLACE PACKAGE BODY dept_emp_pkg
IS
        PROCEDURE show_dept_detail
        ( vdeptno IN dept.deptno%TYPE)
        IS
        BEGIN
                FOR i IN (select * from dept where deptno = vdeptno)
                LOOP
                        dbms_output.put_line(' Department No. : ' || i.deptno ||' Department Name : '||
                        i.dname || ' Location : '|| i.loc);
                        dbms_output.put_line('--------------------------------------------------');
                        dbms_output.put_line('Empno      Name       Designation      Salary');
                        dbms_output.put_line('--------------------------------------------------');
                END LOOP;
                FOR j IN (select * from emp where deptno = vdeptno order by empno)
                LOOP
                        dbms_output.put_line(j.empno ||'        '||j.ename||'       '||j.job||'       '||j.sal);
                END LOOP;
                dbms_output.put_line('--------------------------------------------------');
                FOR k IN (select sum(sal) tot, max(sal) highsal, min(sal) lowsal from emp where deptno =
                vdeptno)
                LOOP
                        dbms_output.put_line('                  Total Salary :        '||k.tot);
                        dbms_output.put_line('                  Total Salary :        '||k.highsal);
                        dbms_output.put_line('                  Total Salary :        '||k.lowsal);
                END LOOP;
                dbms_output.put_line('--------------------------------------------------');
        END;

        FUNCTION emp_exists
        ( vempno IN emp.empno%TYPE)
        RETURN boolean
        IS
                x number;
        BEGIN
                select count(*) into x
                from emp
        where empno = vempno;

        IF x=1 THEN
            return TRUE;
        ELSE
            return FALSE;
        END IF;
    END;

    FUNCTION Gen_Letter_no
    (vdeptno Letter.deptno%TYPE, year_flag char, subject_ref varchar2)
    RETURN varchar2
    IS
        drec Letter%ROWTYPE;
        ans varchar2(100);
        ref char;
        new_no number(4);
        fin_year varchar(10);
    BEGIN
        select * into drec from Letter where deptno = vdeptno;
            ans := drec.dept_name;
        IF year_flag = 'Y' THEN
            ref := to_char(SUBSTR(subject_ref,1,1));
            IF TO_CHAR(sysdate,'MM') > '03' THEN
                fin_year := TO_CHAR(sysdate,'YY')||'-'|| TO_CHAR(TO_NUMBER
                (TO_CHAR(sysdate,'YY'))+1);
            ELSE
                fin_year := TO_CHAR(TO_NUMBER(TO_CHAR(sysdate,'YY'))-1) ||'-'||
                TO_CHAR(sysdate,'YY');
            END IF;
            IF subject_ref IS NULL THEN
                ans := ans || '/' || fin_year;
            ELSE
                IF ref = 'A' THEN
                    ans := ans || '/' || fin_year || '/' || SUBSTR(subject_ref,2);
                ELSE
                    ans := ans || '/' ||SUBSTR(subject_ref,2)||' / '||fin_year ;
                END IF;
            END IF;
        ELSE
            IF subject_ref IS NOT NULL THEN
                ans := ans || '/' ||SUBSTR(subject_ref,2);
            END IF;
        END IF;
        new_no := drec.last_letter_no + 1;
        ans := ans || '/' ||new_no;
        update Letter set last_letter_no = last_letter_no+1 where deptno = vdeptno;
        return ans;
    END;
END;
/
```

→ Calling program 1

```
DECLARE
	vdeptno dept.deptno%type := :enter_deptno;
BEGIN
	dept_emp_pkg.show_dept_detail(vdeptno);
END;
```

→ Calling program 2

```
DECLARE
	vempno emp.empno%TYPE := :enter_empno;
	ans boolean;
BEGIN
	ans := dept_emp_pkg.emp_exists(vempno);
	IF ans = TRUE THEN
		dbms_output.put_line('Employee Exists..');
	ELSE
		dbms_output.put_line('Employee Not Exists...');
	END IF;
END;
```

→ Calling Program 3

```
begin
	dbms_output.put_line(dept_emp_pkg.Gen_Letter_no(10, 'Y', NULL));
	dbms_output.put_line(dept_emp_pkg.Gen_Letter_no(20, 'Y', 'ASALES'));
	dbms_output.put_line(dept_emp_pkg.Gen_Letter_no(20, 'Y', 'BSALES'));
	dbms_output.put_line(dept_emp_pkg.Gen_Letter_no(10, NULL, NULL));
	dbms_output.put_line(dept_emp_pkg.Gen_Letter_no(10, NULL, 'ATENDER'));
	dbms_output.put_line(dept_emp_pkg.Gen_Letter_no(10, NULL, 'BTENDER'));
end;
```

5. Enter employee details and insert the data into emp table. Write all possible exceptions.

```
DECLARE
        vempno emp.empno%TYPE;
        vename emp.ename%TYPE;
        vjob emp.job%TYPE;
        vmgr emp.mgr%TYPE;
        vhiredate emp.hiredate%TYPE;
        vsal emp.sal%TYPE;
        vcomm emp.comm%TYPE;
        vdeptno emp.deptno%TYPE;
        flag number := 0;
BEGIN
        vempno := :enter_empno;
        vename := :enter_ename;
        vjob := :enter_job;
        vmgr := :enter_mgr;
        vhiredate := :enter_hiredate;
        vsal := :enter_sal;
        vcomm := :enter_comm;
        vdeptno := :enter_deptno;

        select 1 into flag
        from dept
        where deptno = vdeptno;

        select 2 into flag
        from emp
        where empno = vmgr;

        insert into emp(empno,ename,job,mgr,hiredate,sal,comm,deptno) values( vempno, vename, vjob,
        vmgr, vhiredate, vsal, vcomm, vdeptno);

        dbms_output.put_line('Record Successfully Inserted..');

EXCEPTION
        WHEN no_data_found THEN
                IF flag = 0 THEN
                        dbms_output.put_line('ERROR := You have entered Invalid Department Number..'||
                        vdeptno);
                ELSIF flag = 1 THEN
                        dbms_output.put_line('ERROR := You have entered Invalid MGR Number..'|| vmgr);
                END IF;
        WHEN dup_val_on_index THEN
                dbms_output.put_line('ERROR := Employee Number Already Exists..'|| vempno);
        WHEN others THEN
                dbms_output.put_line(SQLCODE || '*' || SQLERRM);
END;
/
```

6. Enter employee details and insert the data into emp table with all possible validations. (Validations: deptno should be valid & mgr should be valid. Also validate salary of the employee should not be more than that of his/her manager's salary. Write user-defined exception for validation of salary). Also, write all possible exceptions.

```
DECLARE
        vempno emp.empno%TYPE;
        vename emp.ename%TYPE;
        vjob emp.job%TYPE;
        vmgr emp.mgr%TYPE;
        vhiredate emp.hiredate%TYPE;
        vsal emp.sal%TYPE;
        vcomm emp.comm%TYPE;
        vdeptno emp.deptno%TYPE;
        flag number := 0;
        msal emp.sal%TYPE;
        invalid_sal EXCEPTION;
BEGIN
        vempno := :enter_empno;
        vename := :enter_ename;
        vjob := :enter_job;
        vmgr := :enter_mgr;
        vhiredate := :enter_hiredate;
        vsal := :enter_sal;
        vcomm := :enter_comm;
        vdeptno := :enter_deptno;

        select 1 into flag
        from dept
        where deptno = vdeptno;

        select 2,sal into flag,msal
        from emp
        where empno = vmgr;

        IF msal < vsal THEN
                RAISE invalid_sal;
        END IF;

        insert into emp(empno,ename,job,mgr,hiredate,sal,comm,deptno) values (vempno, vename, vjob,
        vmgr, vhiredate, vsal, vcomm, vdeptno);

        dbms_output.put_line('Record Successfully Inserted..');

EXCEPTION
        WHEN no_data_found THEN
                IF flag = 0 THEN
                        dbms_output.put_line('ERROR := You have entered Invalid Department Number..'||
                        vdeptno);
                ELSIF flag = 1 THEN
                        dbms_output.put_line('ERROR := You have entered Invalid MGR Number..'|| vmgr);
                END IF;
        WHEN dup_val_on_index THEN
                dbms_output.put_line('ERROR := Employee Number Already Exists..'|| vempno);
        WHEN invalid_sal THEN
                dbms_output.put_line('ERROR := Employee salary should be less than their manager..His
                mgr salary is = '|| msal);
        WHEN others THEN
                dbms_output.put_line(SQLCODE || '*' || SQLERRM);
END;
/
```

7. Write a trigger, which will keep track of INSERT, UPDATE & DELETE operations on emp table and store username, date & the name of the event in another table called EMPLOG (create EMPLOG having columns *user*, *date*, *event*, where the event will contain data like 'BEFORE INSERT', 'AFTER DELETE' etc.).

→ Create Table EMPLOG

```
create table emplog(
        username varchar2(30),
        udate date,
        event varchar2(30)
);
```

→ Trigger

```
CREATE OR REPLACE TRIGGER emplog_trigger
AFTER INSERT OR DELETE OR UPDATE ON emp
DECLARE
BEGIN
        IF INSERTING THEN
                insert into emplog values(user,to_char(sysdate,'dd-mm-yyy hh24:mi:ss'),'AFTER INSERT');
        ELSIF UPDATING THEN
                insert into emplog values(user,to_char(sysdate,'dd-mm-yyy hh24:mi:ss'),'AFTER UPDATE');
        ELSIF DELETING THEN
                insert into emplog values(user,to_char(sysdate,'dd-mm-yyy hh24:mi:ss'),'AFTER DELETE');
        END IF;
END;
/
```

8. Write a trigger, which will keep track of INSERT, UPDATE & DELETE operations on dept table and store username, date, event and the data (new and/or updated and/or deleted, whichever is applicable) in another table called DEPTLOG (create DEPTLOG having columns user, date, event, olddeptno, olddname, oldloc, newdeptno, newdname, newloc).

→ Create Table DEPTLOG

```
create table deptlog(
        username varchar2(30),
        udate varchar2(30),
        event varchar2(30),
        olddeptno number(2),
        olddname varchar2(14),
        oldloc varchar2(13),
        newdeptno number(2),
        newdname varchar2(14),
        newloc varchar2(13)
);
```

→ Trigger

```
CREATE OR REPLACE TRIGGER deptlog_trigger
AFTER INSERT OR DELETE OR UPDATE ON dept
FOR EACH ROW
DECLARE
BEGIN
        IF INSERTING THEN
                insert into deptlog values(user,to_char(sysdate,'dd-mm-yyy hh24:mi:ss'),'AFTER INSERT',
                :OLD.deptno,:OLD.dname,:OLD.loc,:NEW.deptno,:NEW.dname,:NEW.loc);
        ELSIF UPDATING THEN
                insert into deptlog values(user,to_char(sysdate,'dd-mm-yyy hh24:mi:ss'),'AFTER UPDATE',
                :OLD.deptno,:OLD.dname,:OLD.loc,:NEW.deptno,:NEW.dname,:NEW.loc);
        ELSIF DELETING THEN
                insert into deptlog values(user,to_char(sysdate,'dd-mm-yyy hh24:mi:ss'),'AFTER DELETE',
                :OLD.deptno,:OLD.dname,:OLD.loc,:NEW.deptno,:NEW.dname,:NEW.loc);
        END IF;
END;
/
```

9. Write a trigger, which will allow the user to perform any updation to EMP table only during office time (office timings are 9 am to 9 pm from Monday to Saturday).

```
CREATE OR REPLACE TRIGGER check_office_time
BEFORE INSERT OR DELETE OR UPDATE ON emp
DECLARE
        found_sunday EXCEPTION;
        found_no_office_time EXCEPTION;
BEGIN
        IF to_char(sysdate,'DY') = 'SUN' THEN
                RAISE found_sunday;
        END IF;
        IF (to_char(sysdate, 'HH24') < 9) OR (to_char(sysdate, 'HH24') > 21 ) THEN
                RAISE found_no_office_time;
        END IF;
EXCEPTION
        WHEN found_sunday THEN
                raise_application_error(-20001,'Today is Sunday. So you can not Edit EMP table....');
        WHEN found_no_office_time THEN
                raise_application_error(-20002,'No Office Time...You can edit in Office time (9AM to 9PM)
                Only....');
END;
/
```