**SIDDHARTH RAMAKRISHNAN\_142897 MODULE-1 ORACLE\_PL/SQL\_ASSIGNMENTS**

**LAB - 1: INTRODUCTION TO DATA DICTIONARY**

**1.1 Get the details of all the database objects and their types created by the current user.**

**ANSWER:** select object\_name,object\_type from user\_objects;

**1.2 Get the details of all the table names owned by current user**

**ANSWER:** Select \* from tab;

**1.3 Get the details of table names and corresponding column names**

**ANSWER:** select object\_name,object\_type from user\_objects;

**1.4 Get the details of column names and corresponding constraint names**

**ANSWER:** select column\_name,constraint\_name from user\_tab\_columns u join all\_constraints a on a.table\_name=u.table\_name;

**1.5 Get the details of the constraints and corresponding table name.**

**ANSWER:** select constraint\_name,a.table\_name from user\_tab\_columns u join all\_constraints a on a.table\_name=u.table\_name;

**1.6 Get the details of all the View names and corresponding Text of the same.**

**ANSWER:** select view\_name,text from all\_views;

**1.7 Get the details of all the Sequence names and their last numbers reached so far.**

**ANSWER:** select sequence\_name,last\_number from all\_sequences;

**1.8 Get the details of all the Synonym names and their parent object names.**

**ANSWER:** select synonym\_name,owner from all\_synonyms;

**1.9 Get the list of all the Index names**

**ANSWER:** select index\_name from all\_indexes;

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

**LAB - 2: INTRODUCTION TO PL/SQL AND CURSORS**

**2.1 Identify the problems (if any) in the below declarations:**

**DECLARE**

**V\_Sample1 NUMBER (2);**

**V\_Sample2 CONSTANT NUMBER (2) ;**

**V\_Sample3 NUMBER (2) NOT NULL ;**

**V\_Sample4 NUMBER (2):= 50;**

**V\_Sample5 NUMBER (2) DEFAULT 25;**

**ANSWER:**

1) V\_Sample2 CONSTANT NUMBER(2); [Should be assigned a Value]

2) V\_Sample3 NUMBER(2) NOT NULL; [Not null must have initialization assignment]

**2.2 The following PL/SQL block is incomplete.**

**Modify the block to achieve requirements as stated in the comments in the block.**

**DECLARE**

**var\_num1 NUMBER := 5;**

**BEGIN**

**DECLARE --inner block**

**var\_num1 NUMBER := 10;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Value for var\_num1:' ||var\_num1);**

**--Can outer block variable (var\_num1) be printed here.If Yes,Print the same.**

**END;**

**--Can inner block variable(var\_num1) be printed here.If Yes,Print the same.**

**END;**

**ANSWER:**

<<outer\_block>>

DECLARE

var\_num1 NUMBER := 5;

BEGIN

<<inner\_block>>

DECLARE

var\_num1 NUMBER := 10;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Value for var\_num1:' ||var\_num1);

DBMS\_OUTPUT.PUT\_LINE('Value for outer var\_num1:' ||outer\_block.var\_num1);

END inner\_block;

-- INNER BLOCK VARIABLE IS OUT OF SCOPE. IT CANNOT BE ACCESSED HERE.

END outer\_block;

**2.3. Write a PL/SQL block to retrieve all staff (code, name, salary) under specific department number and display the result. (Note: The Department\_Code will be accepted from user. Cursor to be used.)**

**ANSWER:**

DECLARE

V\_DCODE STAFF\_MASTERS.DEPT\_CODE%TYPE:=&DCODE;

CURSOR C IS

SELECT STAFF\_CODE,STAFF\_NAME,STAFF\_SAL FROM STAFF\_MASTERS WHERE DEPT\_CODE = V\_DCODE;

BEGIN

FOR R IN C

LOOP

DBMS\_OUTPUT.PUT\_LINE('CODE : '||R.STAFF\_CODE ||' NAME : ' ||R.STAFF\_NAME ||'SALARY : ' ||R.STAFF\_SAL);

END LOOP;

END;

**2.4. Write a PL/SQL block to increase the salary by 30 % or 5000 whichever minimum for a given Department\_Code.**

**ANSWER:**

DECLARE

V\_DCODE STAFF\_MASTERS.DEPT\_CODE%TYPE:=&DCODE;

V\_MIN NUMBER;

V\_MAX NUMBER :=5000;

CURSOR C IS

SELECT DEPT\_CODE,STAFF\_SAL FROM STAFF\_MASTERS;

BEGIN

FOR R IN C

LOOP

IF R.DEPT\_CODE = V\_DCODE

V\_MIN:=R.STAFF\_SAL\*.3;

IF V\_MIN>=V\_MAX THEN

UPDATE STAFF\_MASTERS SET STAFF\_SAL = R.STAFF\_SAL+V\_MAX WHERE DEPT\_CODE=V\_DCODE;

ELSE

UPDATE STAFF\_MASTERS SET STAFF\_SAL = R.STAFF\_SAL+V\_MIN WHERE DEPT\_CODE=V\_DCODE;

V\_MIN:=0;

END IF;

END IF;

END LOOP;

END;

**2.5. Write a PL/SQL block to generate the following report for a given Department code**

**Student\_Code Sudent\_Name Subject1 Subject2 Subject3 Total Percentage Grade**

**For Grade:**

**Student should pass in each subject individually (pass marks 60).**

**Percent >= 80 then grade= A**

**Percent >= 70 and < 80 then grade= B**

**Percent >= 60 and < 70 then grade= C**

**Else D**

**ANSWER:**

DECLARE

V\_DCODE STUDENT\_MASTERS.DEPT\_CODE%TYPE=&DCODE;

V\_GRADE VARCHAR2(2);

CURSOR C IS

SELECT S1.DEPT\_CODE,S1.STUDENT\_CODE AS SCODE,STUDENT\_NAME,SUBJECT1,SUBJECT2,SUBJECT3,SUBJECT1+SUBJECT2+SUBJECT3 AS TOTAL,ROUND((SUBJECT1+SUBJECT2+SUBJECT3)/3),2) AS PERC

FROM STUDENT\_MASTERS S1 JOIN STUDENT\_MARKS S2 ON S1.STUDENT\_CODE=S2.STUDENT\_CODE

WHERE SUBJECT1>=60 AND SUBJECT2>=60 AND SUBJECT3>=60;

BEGIN

FOR R IN C

LOOP

IF R.PERC>=80 THEN

V\_GRADE:='A';

ELSIF R.PERC>=70 AND R.PERC<80 THEN

V\_GRADE:='B';

ELSIF R.PERC>=60 AND R.PERC<70 THEN

V\_GRADE:='C'

ELSE

V\_GRADE:='D';

END IF;

DBMS\_OUTPUT.PUT\_LINE('STUDENT CODE : '||R.SCODE||' STUDENT NAME : '||R.STUDENT\_NAME||

' SUBJECT 1 : ' ||R.SUBJECT1||' SUBJECT 2 : '|| R.SUBJECT2||

' SUBJECT 3 : ' ||R.SUBJECT3|| ' TOTAL : ' ||R.TOTAL||' PERCENTAGE : '

||R.PERC||'GRADE : '||V\_GRADE);

END LOOP;

END;

**2.6. Write a PL/SQL block to retrieve the details of the staff belonging to a particular department. Department code should be passed as a parameter to the cursor.**

**ANSWER:**

DECLARE

CURSOR C(DNO NUMBER) IS SELECT \* FROM STAFF\_MASTERS WHERE DEPT\_CODE=DNO;

BEGIN

FOR R IN C(10)

LOOP

DBMS\_OUTPUT.PUT\_LINE('STAFF CODE : ' || R.STAFF\_CODE||

'STAFF NAME : ' || R.STAFF\_NAME||

'DESIGN CODE : '|| R.DESIGN\_CODE||

'DEPT CODE : ' || R.DEPT\_CODE||

'STAFF DOB : ' || R.STAFF\_DOB||

'HIRE DATE : ' ||R.HIREDATE||

'MGR CODE : ' ||R.MGR\_CODE||

'STAFF SALARY : ' ||R.STAFF\_SAL||

'STAFF ADDRESS : '|| R.STAFF\_ADDRESS);

END LOOP;

END;

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

**LAB - 3: EXCEPTION HANDLING AND DYNAMIC SQL**

**3.1 Modify the programs created in Lab2 to implement Exception Handling**

**ANSWER:**

**(2.3)**

DECLARE

V\_DCODE STAFF\_MASTERS.DEPT\_CODE%TYPE:=&DCODE;

CURSOR C IS

SELECT STAFF\_CODE,STAFF\_NAME,STAFF\_SAL FROM STAFF\_MASTERS WHERE DEPT\_CODE = V\_DCODE;

BEGIN

FOR R IN C

LOOP

DBMS\_OUTPUT.PUT\_LINE('CODE : '||R.STAFF\_CODE ||' NAME : ' ||R.STAFF\_NAME

||'SALARY : ' ||R.STAFF\_SAL);

END LOOP;

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID INPUT');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO DATA');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUPUT.PUT\_LINE(' MORE THAN ONE ROW ');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('OTHER ERROR');

END;

**(2.4)**

DECLARE

V\_DCODE STAFF\_MASTERS.DEPT\_CODE%TYPE:=&DCODE;

V\_MIN NUMBER;

V\_MAX NUMBER:=5000;

CURSOR C IS

SELECT DEPT\_CODE,STAFF\_SAL FROM STAFF\_MASTERS;

BEGIN

FOR R IN C

LOOP

V\_MIN:=R.STAFF\_SAL\*.3;

IF V\_MIN>=V\_MAX THEN

UPDATE STAFF\_MASTERS SET STAFF\_SAL = R.STAFF\_SAL+V\_MAX WHERE DEPT\_CODE=V\_DCODE;

ELSE

UPDATE STAFF\_MASTERS SET STAFF\_SAL = R.STAFF\_SAL+V\_MIN WHERE DEPT\_CODE=V\_DCODE;

V\_MIN:=0;

END IF;

END LOOP;

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID INPUT');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO DATA');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE(' MORE THAN ONE ROW ');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('OTHER ERROR');

END;

**(2.5)**

DECLARE

V\_DCODE STUDENT\_MASTERS.DEPT\_CODE%TYPE=&DCODE;

V\_GRADE VARCHAR2(2);

CURSOR C IS

SELECT S1.DEPT\_CODE,S1.STUDENT\_CODE AS SCODE,STUDENT\_NAME,SUBJECT1,SUBJECT2,SUBJECT3,

SUBJECT1+SUBJECT2+SUBJECT3 AS TOTAL,ROUND((SUBJECT1+SUBJECT2+SUBJECT3)/3),2) AS PERC FROM STUDENT\_MASTERS S1 JOIN STUDENT\_MARKS S2 ON S1.STUDENT\_CODE=S2.STUDENT\_CODE

WHERE SUBJECT1>=60 AND SUBJECT2>=60 AND SUBJECT3>=60;

BEGIN

FOR R IN C

LOOP

IF R.PERC>=80 THEN

V\_GRADE:='A';

ELSIF R.PERC>=70 AND R.PERC<80 THEN

V\_GRADE:='B';

ELSIF R.PERC>=60 AND R.PERC<70 THEN

V\_GRADE:='C';

ELSE

V\_GRADE:='D';

END IF;

DBMS\_OUTPUT.PUT\_LINE('STUDENT CODE : '||R.SCODE||' STUDENT NAME : '||R.STUDENT\_NAME||

' SUBJECT 1 : ' ||R.SUBJECT1||' SUBJECT 2 : '|| R.SUBJECT2||

' SUBJECT 3 : ' ||R.SUBJECT3|| ' TOTAL : ' ||R.TOTAL||' PERCENTAGE : '

||R.PERC||'GRADE : '||V\_GRADE);

END LOOP;

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID INPUT');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO DATA');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUPUT.PUT\_LINE(' MORE THAN ONE ROW ');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('OTHER ERROR');

END;

**(2.6)**

DECLARE

V\_DCODE STAFF\_MASTERS.DEPT\_CODE%TYPE:=&DCODE;

CURSOR C IS

SELECT \* FROM STAFF\_MASTERS WHERE DEPT\_CODE=V\_DCODE;

BEGIN

FOR R IN C

LOOP

DBMS\_OUTPUT.PUT\_LINE('STAFF CODE : ' || R.STAFF\_CODE||

'STAFF NAME : ' || R.STAFF\_NAME||

'DESIGN CODE : '|| R.DESIGN\_CODE||

'DEPT CODE : ' || R.DEPT\_CODE||

'STAFF DOB : ' || R.STAFF\_DOB||

'HIRE DATE : ' ||R.HIREDATE||

'MGR CODE : ' ||R.MGR\_CODE||

'STAFF SALARY : ' ||R.STAFF\_SAL||

'STAFF ADDRESS : '|| R.STAFF\_ADDRESS);

END LOOP;

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID INPUT');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO DATA');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUPUT.PUT\_LINE(' MORE THAN ONE ROW ');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('OTHER ERROR');

END;

**3.2 The following PL/SQL block attempts to calculate bonus of staff for a given MGR\_CODE. Bonus is to be considered as twice of salary. Though Exception Handling has been implemented but block is unable to handle the same.**

**Debug and verify the current behavior to trace the problem.**

**DECLARE**

**V\_BONUS V\_SAL%TYPE;**

**V\_SAL STAFF\_MASTER.STAFF\_SAL%TYPE;**

**BEGIN**

**SELECT STAFF\_SAL INTO V\_SAL**

**FROM STAFF\_MASTER**

**WHERE MGR\_CODE=100006;**

**V\_BONUS:=2\*V\_SAL;**

**DBMS\_OUTPUT.PUT\_LINE('STAFF SALARY IS ' || V\_SAL);**

**DBMS\_OUTPUT.PUT\_LINE('STAFF BONUS IS ' || V\_BONUS);**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('GIVEN CODE IS NOT VALID.ENTER VALID CODE');**

**END;**

**ANSWER:**

DECLARE

V\_BONUS STAFF\_MASTERS.STAFF\_SAL%TYPE; -- [[V\_BONUS STAFF\_MASTERS.STAFF\_SAL%TYPE [TYPE MISMATCH]]

V\_SAL STAFF\_MASTER.STAFF\_SAL%TYPE;

BEGIN

SELECT STAFF\_SAL INTO V\_SAL

FROM STAFF\_MASTER

WHERE MGR\_CODE=100006;

V\_BONUS:=2\*V\_SAL;

DBMS\_OUTPUT.PUT\_LINE('STAFF SALARY IS ' || V\_SAL);

DBMS\_OUTPUT.PUT\_LINE('STAFF BONUS IS ' || V\_BONUS);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('GIVEN CODE IS NOT VALID.ENTER VALID CODE');

WHEN TOO\_MANY\_ROWS THEN -- [[TOO MANY ROWS EXCEPTION ADDED]]

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS');

END;

**3.3 Rewrite the above block to achieve the requirement**.

**ANSWER:**

DECLARE

CURSOR cur\_3\_2 is SELECT STAFF\_SAL FROM STAFF\_MASTERs WHERE MGR\_CODE=100006;

V\_SAL STAFF\_MASTERs.STAFF\_SAL%TYPE;

V\_BONUS V\_SAL%TYPE;

BEGIN

FOR vcur in cur\_3\_2

LOOP

V\_BONUS:=2\*vcur.staff\_SAL;

DBMS\_OUTPUT.PUT\_LINE('STAFF SALARY IS ' || vcur.staff\_SAL);

DBMS\_OUTPUT.PUT\_LINE('STAFF BONUS IS ' || V\_BONUS);

END LOOP;

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID INPUT');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO DATA');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE(' MORE THAN ONE ROW ');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('OTHER ERROR');

END;

**3.4 Predict the output of the following block ? What corrections would be needed to make it more efficient?**

**ANSWER:**

BEGIN

DECLARE

fname emp.ename%TYPE;

BEGIN

SELECT ename INTO fname FROM emp WHERE 1=2;

DBMS\_OUTPUT.PUT\_LINE('This statement will print');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some inner block error');

END;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found in fname');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some outer block error');

END;

CORRECTION :

BEGIN

DECLARE

fname emp.ename%TYPE;

BEGIN

SELECT ename INTO fname FROM emp WHERE 1=2;

DBMS\_OUTPUT.PUT\_LINE('This statement will print');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('DATA NOT FOUND ' ); -- EXCEPTION ADDED

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some inner block error');

END;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found in fname');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some outer block error');

END;

**3.5. Debug the above block to trace the flow of control.**

**Additionally one can make appropriate changes in Select statement defined in the block to check the flow.**

**ANSWER:**

BEGIN

DECLARE

FNAME EMP.ENAME%TYPE;

CURSOR CUR3\_5 IS SELECT ENAME FROM EMP ;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('This statement will print');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some inner block error');

END;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found in fname');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Some outer block error');

END;

**3.6. Write a PL/SQL program to check for the commission for an employee no 7369. If no Commission exists, then display the error message. Use Exceptions.**

**ANSWER:**

DECLARE

VCOMM EMPLOYEES.COMMISSION\_PCT%TYPE;

COMM\_NULL EXCEPTION;

BEGIN

SELECT COMMISSION\_PCT INTO VCOMM FROM EMPLOYEES WHERE EMPLOYEE\_ID=100;

IF (VCOMM IS NULL) THEN

RAISE COMM\_NULL;

ELSE

DBMS\_OUTPUT.PUT\_LINE('COMM IS'||VCOMM);

END IF;

EXCEPTION

WHEN COMM\_NULL THEN

DBMS\_OUTPUT.PUT\_LINE('COMM ISNULL');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO DTA FOUND');

END;

**3.7. Write a PL/SQL block to drop any user defined table.**

**ANSWER:**

DECLARE

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE &TABLE\_NAME';

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID TABLE NAME');

END;

**LAB - 4: DATABASE PROGRAMMING**

**4.1 Write a PL/SQL block to find the maximum salary of the staff in the given department. Note: Department code should be passed as parameter to the cursor.**

**ANSWER:**

declare

max staff\_masters.staff\_sal%type;

cursor cur(code staff\_masters.dept\_code%type) is

select max(staff\_sal) max from staff\_masters

group by dept\_code having dept\_code=code;

vcur cur%rowtype;

begin

open cur(&code);

loop

fetch cur into vcur;

exit when cur%notfound;

dbms\_output.put\_line(vcur.max);

end loop;

exception

when no\_data\_found then

dbms\_output.put\_line(‘No data Found’);

when too\_many\_rows then

dbms\_output.put\_line(‘More than one row’);

when others then

dbms\_output.put\_line(‘Other errors’);

end;

**4.2. Write a function to compute age. The function should accept a date and return age in years.**

**ANSWER:**

create or replace function calc\_age(dob date)

return number is

age number(2):=0;

begin

age:=round(sysdate-dob)/(30\*12);

return age;

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end;

/\*To execute\*/

declare

age number(2);

dob date:=to\_date('27-MAY-1996');

begin

age:=calc\_age(dob);

dbms\_output.put\_line('Age: '||age);

end;

**4.3. Write a procedure that accept staff code and update staff name to Upper case. If the staff name is null raise a user defined exception.**

**ANSWER:**

create or replace procedure update\_proc(staffCode in number) as

staffName varchar(20);

exp1 EXCEPTION;

begin

select staff\_name into staffName from staff\_masters

where staff\_code=staffCode;

if staffName is NULL then

raise exp1;

end if;

update staff\_masters set staff\_name=UPPER(staff\_name) where staff\_code=staffCode;

EXCEPTION

when exp1 then

dbms\_output.put\_line('staff name is null');

end update\_proc;

/\*To execute\*/

declare

staff\_name varchar(20);

begin

update\_proc(100002);

end;

**4.4 Write a procedure to find the manager of a staff. Procedure should return the following – Staff\_Code, Staff\_Name, Dept\_Code and Manager Name.**

**ANSWER:**

create or replace procedure disp(mgrCode number) as

staffdetails staff\_masters%rowtype;

cursor r is

Select distinct e.staff\_code as staffNo, e.staff\_name as staff, dept\_name, (select staff\_name from staff\_masters where staff\_code=mgrCode) as manager from staff\_masters e, staff\_masters m, department\_masters d where e.mgr\_code=mgrCode and e.dept\_code=d.dept\_code;

begin

for c in r

loop

dbms\_output.put\_line(c.staffNo||' '||c.staff||' '||c.dept\_name||' '||c.manager);

end loop;

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end;

/\*To Execute\*/

exec disp(100005);

**4.5. Write a function to compute the following. Function should take Staff\_Code and return the cost to company.**

**DA = 15% Salary, HRA= 20% of Salary, TA= 8% of Salary.**

**Special Allowance will be decided based on the service in the company.**

**< 1 Year Nil**

**>=1 Year< 2 Year 10% of Salary**

**>=2 Year< 4 Year 20% of Salary**

**>4 Year 30% of Salary**

**ANSWER:**

create or replace function cost(staffcode number)

return number is

total number;

cursor c is select staff\_sal, (0.15\*staff\_sal) as da, (0.2\*staff\_sal) as hra, (0.08\*staff\_sal) as ta,

case when round(sysdate-to\_date(hiredate))<1 then 0

when round(sysdate-to\_date(hiredate))>=1 and round(sysdate-to\_date(hiredate))<2 then (0.1\*staff\_sal)

when round(sysdate-to\_date(hiredate))>=2 and round(sysdate-to\_date(hiredate))<4 then (0.2\*staff\_sal)

when round(sysdate-to\_date(hiredate))>4 then (0.3\*staff\_sal) end as sa

from staff\_masters where staff\_code=staffcode;

begin

for r in c

loop

total:=r.staff\_sal+r.da+r.hra+r.ta+r.sa;

return total;

end loop;

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end;

/\* To Execute\*/

declare

total number(9,2);

begin

total:=cost(100006);

dbms\_output.put\_line('total cost='||total);

end;

**4.6. Write a procedure that displays the following information of all staff**

**Staff\_Name Department Name Designation Salary Status**

**Note: - Status will be (Greater, Lesser or Equal) respective to average salary of their own department. Display an error message Staff\_Master table is empty if there is no matching record.**

**ANSWER:**

create or replace procedure staff(deptno in number)

as

staff\_detail staff\_masters%rowtype;

cursor c is select staff\_name, dept\_name, design\_name, staff\_sal, (select avg(staff\_sal) from staff\_masters where dept\_code=deptno) as avg,

case when staff\_sal<(select avg(staff\_sal) from staff\_masters where dept\_code=deptno) then 'lesser'

when staff\_sal=(select avg(staff\_sal) from staff\_masters where dept\_code=deptno) then 'equal'

when staff\_sal>(select avg(staff\_sal) from staff\_masters where dept\_code=deptno) then 'greater'

end as status

from staff\_masters s, department\_masters d, designation\_masters n

where s.dept\_code=d.dept\_code and s.dept\_code=deptno and s.design\_code=n.design\_code;

begin

for r in c

loop

dbms\_output.put\_line(r.staff\_name||' '||r.dept\_name||' '||r.design\_name||' '||r.staff\_sal||' '||r.status);

end loop;

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end;

/\* To Execute \*/

exec staff(10);

**4.7. Write a procedure that accept Staff\_Code and update the salary and store the old salary details in Staff\_Master\_Back (Staff\_Master\_Back has the same structure without any constraint) table.**

**Exp < 2 then no Update**

**Exp > 2 and < 5 then 20% of salary**

**Exp > 5 then 25% of salary**

**ANSWER:**

create or replace procedure proc7(code staff\_masters.staff\_code%type) is

sal staff\_masters.staff\_sal%type;

updatesal number;

exp number;

begin

select staff\_sal,round(months\_between(sysdate,hiredate)/12,0) into sal,exp

from staff\_masters where staff\_code=code;

if (exp <2) then

dbms\_output.put\_line('You are not eligible to update');

updatesal:=sal;

elsif (exp>=2 and exp<5) then

updatesal:=sal+sal\*0.2;

else

updatesal:=sal+sal\*0.25;

end if;

update staff\_masters set staff\_sal=updatesal where staff\_code=code;

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end proc7;

/\*To execute\*/

exec proc7(100005);

**4.8. Create a procedure that accepts the book code as parameter from the user. Display the details of the students/staff that have borrowed that book and has not returned the same. The following details should be displayed**

**Student/Staff Code Student/Staff Name Issue Date Designation Expected Ret\_Date**

**ANSWER:**

create or replace procedure proc8(code book\_transactions.book\_code%type) is

scode1 number;

scode2 number;

name varchar2(30);

bid date;

bart date;

begin

select staff\_code,student\_code into scode1,scode2 from book\_transactions where book\_code=code;

if(scode1 is not null) then

select s.staff\_name,b.book\_issue\_date,b.book\_actual\_return\_date into name,bid,bart

from staff\_masters s join book\_transactions b

on s.staff\_code=b.staff\_code where s.staff\_code=scode1;

dbms\_output.put\_line(scode1);

elsif (scode2 is not null) then

select s.student\_name,b.book\_issue\_date,b.book\_actual\_return\_date into name,bid,bart

from student\_masters s join book\_transactions b

on s.student\_code=b.student\_code where s.student\_code=scode2;

dbms\_output.put\_line(scode2);

else

dbms\_output.put\_line('no data found');

end if;

dbms\_output.put\_line(name);

dbms\_output.put\_line(bid);

dbms\_output.put\_line(bart);

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end proc8;

/\*To execute\*/

exec proc8(10000009);

**4.9. Write a package which will contain a procedure and a function.**

**Function: This function will return years of experience for a staff. This function will take**

**the hiredate of the staff as an input parameter. The output will be rounded to the nearest**

**year (1.4 year will be considered as 1 year and 1.5 year will be considered as 2 year).**

**Procedure: Capture the value returned by the above function to calculate the additional**

**allowance for the staff based on the experience.**

**Additional Allowance = Year of experience x 3000.**

**Calculate the additional allowance and store Staff\_Code, Date of Joining, and Experience**

**in years and additional allowance in Staff\_Allowance table.**

**ANSWER:**

/\*Create Package\*/

CREATE OR REPLACE PACKAGE COMM\_PKG

IS

V\_CODE NUMBER;

PROCEDURE FP\_ALL(V\_CODE NUMBER);

END COMM\_PKG;

/

/\*Staff Allowance Table\*/

create table staff\_allowance(staff\_code number,hiredate date,exp number,allowance number);

/\*Package Body Creation\*/

CREATE OR REPLACE PACKAGE BODY COMM\_PKG

IS

FUNCTION F\_DATE(V\_CODE NUMBER)

RETURN NUMBER

IS

V\_AGE NUMBER;

BEGIN

SELECT ROUND(MONTHS\_BETWEEN(SYSDATE,HIREDATE)/12) INTO

V\_AGE FROM STAFF\_MASTERS

WHERE STAFF\_CODE=V\_CODE;

RETURN V\_AGE;

END F\_DATE;

PROCEDURE FP\_ALL(V\_CODE IN NUMBER)

IS

V\_NUM NUMBER;

V\_HIRE DATE;

V\_ALLOW NUMBER;

BEGIN

SELECT HIREDATE INTO V\_HIRE FROM

STAFF\_MASTERS

WHERE STAFF\_CODE=V\_CODE;

V\_NUM:=F\_DATE(V\_CODE);

V\_ALLOW:=V\_NUM\*3000;

INSERT INTO STAFF\_ALLOWANCE VALUES

(V\_CODE,V\_HIRE,V\_NUM,V\_ALLOW);

DBMS\_OUTPUT.PUT\_LINE(V\_ALLOW);

END FP\_ALL;

END COMM\_PKG;

/

/\*Package Execution\*/

DECLARE

BEGIN

COMM\_PKG.FP\_ALL(100001);

END;

/

**4.10. Write a procedure to insert details into Book\_Transaction table. Procedure should accept the book code and staff/student code. Date of issue is current date and the expected return date should be 10 days from the current date. If the expected return date falls on Saturday or Sunday, then it should be the next working day.**

**ANSWER:**

create or replace procedure proc10

(bcode book\_transactions.book\_code%type,stucode book\_transactions.student\_code%type,

stfcode book\_transactions.staff\_code%type) is

rdate date;

begin

rdate:=sysdate+10;

if(to\_char(rdate,'dy')='sat') then

rdate:=sysdate+12;

elsif(to\_char(rdate,'dy')='sun') then

rdate:=sysdate+11;

else

rdate:=sysdate+10;

end if;

insert into book\_transactions values(bcode,stucode,stfcode,sysdate,rdate,null);

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end proc10;

/\*To execute\*/

exec proc10(10000001,1010,null);

/\*To check Table\*/

select \* from book\_transactions;

**4.11. Write a function named ‘get\_total\_records’, to pass the table name as a parameter,**

**and get back the number of records that are contained in the table. Test your function**

**with multiple tables.**

**ANSWER:**

/\*Function Creation\*/

CREATE OR REPLACE FUNCTION

GET\_TOTAL\_RECORDS(V\_TABLE VARCHAR2)

RETURN NUMBER

IS

V\_REC NUMBER;

BEGIN

execute immediate 'SELECT COUNT(\*) FROM ' ||V\_TABLE INTO V\_REC;

RETURN V\_REC;

END;

/\*Function Calling\*/

DECLARE

V\_NUM NUMBER;

BEGIN

V\_NUM:=GET\_TOTAL\_RECORDS('BOOK\_TRANSACTIONS');

DBMS\_OUTPUT.PUT\_LINE(V\_NUM);

END;

**4.12**

**Tune the following Oracle Procedure enabling to gain better performance.**

**Objective:The Procedure should update the salary of an employee and at the same time retrieve the employee's name and new salary into PL/SQL variables.**

**ANSWER:**

CREATE OR REPLACE PROCEDURE update\_salary (emp\_id NUMBER) IS

v\_name VARCHAR2(15);

v\_newsal NUMBER;

BEGIN

UPDATE emp SET sal = sal \* 1.1

WHERE empno = emp\_id;

SELECT ename, sal INTO v\_name, v\_newsal

FROM emp

WHERE empno = emp\_id;

DBMS\_OUTPUT.PUT\_LINE('Emp Name:' || v\_name);

DBMS\_OUTPUT.PUT\_LINE('Ename:' || v\_newsal);

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

END;

/\*To execute\*/

exec update\_salary(7499);

**4.13**

**The following procedure attempts to delete data from table passed as parameter.This procedure has compilation errors. Identify and correct the problem.**

**ANSWER:**

CREATE or REPLACE PROCEDURE gettable(table\_name in varchar2) AS

BEGIN

execute immediate 'DELETE FROM table\_name';

END;

CREATE or REPLACE PROCEDURE gettable(table\_name in varchar2) AS

BEGIN

execute immediate 'DELETE FROM' || table\_name;

END;

/

**4.14**

**Write a procedure which prints the following report using procedure:**

**The procedure should take deptno as user input and appropriately print the emp details.**

**Also display : Number of Employees,Total Salary,Maximum Salary,Average Salary**

**Note: The block should achieve the same without using Aggregate Functions.**

**ANSWER:**

create or replace procedure proc14(dno emp.deptno%type) is

cont number:=0;

tot number;

average number;

maxm number;

cursor cur1 is select ename,job,sal,comm from emp where deptno=dno;

begin

for vcur in cur1

loop

dbms\_output.put\_line('Employee Name :'||vcur.ename);

dbms\_output.put\_line('Employee Job :'||vcur.job);

dbms\_output.put\_line('Employee Salary :'||vcur.sal);

dbms\_output.put\_line('Employee Comission :'||vcur.comm);

dbms\_output.put\_line('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

cont:=cont+1;

select sum(sal),round(avg(sal),2),max(sal) into tot,average,maxm from emp group by dno;

end loop;

dbms\_output.put\_line('Number of Employees :'||cont);

dbms\_output.put\_line('Total Salary :'||tot);

dbms\_output.put\_line('Maximum Salary :' ||maxm);

dbms\_output.put\_line('Average Salary :'||average);

dbms\_output.put\_line('---------------------------------------------');

EXCEPTION

when value\_error then

dbms\_output.put\_line('invalid input');

when no\_data\_found then

dbms\_output.put\_line('no data');

when too\_many\_rows then

dbms\_output.put\_line(' more than one row ');

when others then

dbms\_output.put\_line('other error');

end proc14;

/\*To execute\*/

exec proc14(20);

**4.15: Write a query to view the list of all procedures ,functions and packages from the Data Dictionary.**

**ANSWER:**

select \* from user\_objects where object\_type like

'PROCEDURE' or object\_type like 'FUNCTION' or

object\_type like 'PACKAGE';

**LAB - 8: SQL\*PLUS REPORTS**

**8.1**

**Using Multiple Spacing Techniques**

**Suppose you have more than one column in your ORDER BY clause and wish to insert space when each column’s value changes. Each BREAK command you enter replaces the previous one.**

**Now consider a scenario where you want to do either of the following:**

* **to use different spacing techniques in one report, or**
* **to insert space after the value changes in more than one ordered column**

**Then you must specify “multiple columns” and “actions” in a single BREAK command.**

**ANSWER:**

select deptno,job,ename,sal from emp101 where sal > 1200 order by deptno,job;

BREAK ON deptno SKIP PAGE ON job SKIP 1;

/\*TTITLE is yet to complete\*/

COLUMN DUMMY NOPRINT

COMPUTE AVG OF SAL ON DUMMY

BREAK ON DUMMY SKIP 1

SELECT JOB DUMMY,DEPTNO,JOB,ENAME,SAL

FROM EMP

WHERE SAL >1200 ORDER BY DEPTNO,JOB;

BREAK ON DEPTNO ON JOB

COMPUTE AVG OF SAL ON JOB

/

**8.2 Computing and Printing Subtotals**

**ANSWER:**

COLUMN DEPTNO HEADING 'DEPARTMENT\_ID'

COLUMN ENAME HEADING 'NAME'

COLUMN SAL HEADING 'SALARY'

TTITLE LEFT 'SALES DEPARTMENT PERSONNEL REPORT' SKIP 1 LEFT 'PERFECT WIDGETS' SKIP 1 RIGHT '8-JUL-2017' SKIP 1 RIGHT FORMAT 9 'PAGE:' SQL.PNO

BTITLE LEFT 'COMPANY CONFIDENTIAL'

select deptno,ename,sal from emp;