LAB-1:

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;

CODE:

2.1

DECLARE

V\_Sample1 NUMBER(2);

V\_Sample2 CONSTANT NUMBER(2) :=0;

V\_Sample3 NUMBER(2) NOT NULL :=0;

V\_Sample4 NUMBER(2) := 50;

V\_Sample5 NUMBER(2) DEFAULT 25;

BEGIN

null;

END;

/

2.2)

The following PL/SQL block is incomplete.

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

DECLARE --outer block

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;

CODE:

DECLARE --outer block

var\_num1 NUMBER := 5;

BEGIN

DECLARE --inner block

var\_num1 NUMBER := 10;

BEGIN

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

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

END;

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

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

END;

/

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.)

CODE:

DECLARE

ccode staff\_masters.staff\_code%type;

cname staff\_masters.staff\_name%type;

csal staff\_masters.staff\_sal%type;

cursor c is select staff\_code, staff\_name, staff\_sal from staff\_masters where dept\_code=&k;

BEGIN

open c;

loop

exit when c%notfound;

fetch c into ccode,cname, csal;

dbms\_output.put\_line(ccode || ' ' || cname || ' ' || csal);

END loop;

close c;

END;

/

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

CODE:

DECLARE

ccode staff\_masters.staff\_code%type;

cname staff\_masters.staff\_name%type;

csal staff\_masters.staff\_sal%type;

cursor c is select staff\_code, staff\_name, staff\_sal from staff\_masters where staff\_sal in (select staff\_sal+5000 from staff\_masters);

BEGIN

open c;

loop

exit when c%notfound;

fetch c into ccode,cname, csal;

dbms\_output.put\_line(ccode || ' ' || cname || ' ' || csal);

END loop;

close c;

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

Note: Display suitable error massage if wrong department code has entered and if there is no student in the given department.

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

2.5)

CODE:

DECLARE

scode student\_marks.student\_code%type;

sname student\_masters.student\_name%type;

ssubject1 student\_marks.subject1%type;

ssubject2 student\_marks.subject2%type;

ssubject3 student\_marks.subject3%type;

total number(10);

percentage number(10,2);

grade char(1);

dt department\_masters.dept\_code%type:=&k;

cursor c is select a.student\_code, b.student\_name, a.subject1, a.subject2, a.subject3 from student\_marks a join student\_masters b on a.student\_code=b.student\_code where b.dept\_code=dt;

BEGIN

open c;

loop

exit when c%notfound;

fetch c into scode,sname,ssubject1,ssubject2,ssubject3;

total:=ssubject1+ssubject2+ssubject3;

percentage:=total/3;

if(percentage >= 80)

then grade:='A';

elsif(percentage >= 70 and percentage < 80)

then grade:='B';

elsif(percentage >= 60 and percentage < 70)

then grade:='C';

else

grade:='D';

END if;

dbms\_output.put\_line(scode || ' ' || sname || ' ' || ssubject1 || ' ' || ssubject2 || ' '|| ssubject3 || ' '|| total || ' ' || percentage || ' ' || grade);

END loop;

close c;

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.

CODE:

DECLARE

temp staff\_masters%rowtype;

cursor s(n number) is select \* from staff\_masters where dept\_code=n;

BEGIN

for temp in s(&n)

loop

dbms\_output.put\_line(temp.staff\_code||' ' || temp.staff\_name||' '||temp.dept\_code||' '||temp.staff\_sal||' '||temp.staff\_dob);

END loop;

END;

/

===========================================================================================

LAB-2:

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

CODE:

Set verify off;

Set serveroutput on;

DECLARE

temp staff\_masters%rowtype;

cursor s(n number) is select \* from staff\_masters where dept\_code=n;

invalid exception;

val number(5);

BEGIN

val:=&n;

open s(val);

fetch s into temp;

if s%notfound then

raise invalid;

END if;

loop

fetch s into temp;

dbms\_output.put\_line(temp.staff\_code||' ' || temp.staff\_name||' '||temp.dept\_code||' '||temp.staff\_sal||' '||temp.staff\_dob);

END loop;

exception

when invalid

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

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;

CODE:

3.2

DECLARE

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

V\_BONUS V\_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 too\_many\_rows then

DBMS\_OUTPUT.PUT\_LINE('To many records');

WHEN NO\_DATA\_FOUND THEN

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

END;

/

3.3) Rewrite the above block to achieve the requirement.

CODE:

3.3

DECLARE

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

V\_BONUS V\_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 too\_many\_rows then

DBMS\_OUTPUT.PUT\_LINE('To many records');

WHEN NO\_DATA\_FOUND THEN

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

END;

/

3.4

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

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;

3.4)

CODE:

BEGIN

DECLARE

fname emp.ename%TYPE;

BEGIN

SELECT ename INTO fname

FROM emp;

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;

/

3.5 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.

3.5)

CODE:

BEGIN

DECLARE

fname emp.ename%TYPE;

BEGIN

SELECT ename INTO fname

FROM emp;

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.

3.6)

CODE:

DECLARE

vemp emp.empno%type;

vname emp.ename%type;

vcomm emp.comm%type;

exp exception;

BEGIN

select empno, ename, comm into vemp, vname, vcomm from emp where empno=&num;

if vcomm is null

then

raise exp;

else

dbms\_output.put\_line(vemp||' '||vname||' '||vcomm);

END if;

exception

when exp then

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

END;

/

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

3.7)

CODE:

DECLARE

tname varchar2(20);

BEGIN

tname:=&tname;

execute immediate('drop table '||tname);

END;

=========================================================================================

LAB-3

4.1Write 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.

DECLARE

V\_MAX\_SAL NUMBER(10,2);

V\_CODE NUMBER(2);

CURSOR C\_STAFF IS

SELECT DEPT\_CODE,MAX(STAFF\_SAL) FROM STAFF\_MASTERS GROUP BY DEPT\_CODE;

BEGIN

OPEN C\_STAFF;

LOOP

FETCH C\_STAFF INTO V\_CODE,V\_MAX\_SAL;

DBMS\_OUTPUT.PUT\_LINE(V\_CODE|| ' ' || V\_MAX\_SAL);

END LOOP;

CLOSE C\_STAFF;

END;

/

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

years.

SQL> CREATE OR REPLACE FUNCTION FINDAGE(

DOB IN DATE)

RETURN NUMBER

IS

YRS NUMBER;

BEGIN

YRS:= ROUND((SYSDATE - DOB)/365);

RETURN YRS;

END;

/

SQL> select findage('27-MAY-76') from dual;

FINDAGE('27-MAY-76')

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

42

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.

create or replace procedure toUpperCase

(SNo in staff\_masters.staff\_code%type,

SName out staff\_masters.staff\_name%type)

as

BEGIN

select upper(STAFF\_NAME) into SName from staff\_masters where staff\_code=SNo;

if(sql%notfound) then

raise no\_data\_found;

else

UPDATE staff\_masters

SET staff\_name=SName

WHERE staff\_code=SNo;

END if;

exception

when no\_data\_found then

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

END;

variable name varchar2;

execute toUpperCase(&EnterStaffCode,:name);

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.

create or replace procedure findMGR

(SNo in staff\_masters.staff\_code%type,

SId out staff\_masters.staff\_code%type,

SName out staff\_masters.staff\_name%type,

DCode out staff\_masters.dept\_code%type,

MName out staff\_masters.staff\_name%type)

as

BEGIN

select a.staff\_code,a.staff\_name,a.dept\_code,b.staff\_name as manager into SId,SName,DCode,MName

from staff\_masters a,staff\_masters b

where a.mgr\_code=b.staff\_code and a.staff\_code = SNo;

if(sql%notfound) then

raise no\_data\_found;

END if;

exception

when no\_data\_found then

dbms\_output.put\_line('staff code not found');

END;

variable code number

variable name varchar2

variable dcode number

variable mname varchar2

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

create or replace function CostToComp(code in number) return number

is

sal staff\_masters.staff\_sal%type;

exp number;

BEGIN

select staff\_sal,round(months\_between(sysdate,hiredate)/12) as EXP into sal, exp from staff\_masters where staff\_code=code;

if exp>4 then

sal:=sal+ (0.3\*sal)+(0.15\*sal)+(0.2\*sal)+(0.08\*sal);

elsif exp between 2 and 4 then

sal:=sal+ (0.2\*sal)+(0.15\*sal)+(0.2\*sal)+(0.08\*sal);

elsif exp between 1 and 2 then

sal:=sal+ (0.1\*sal)+(0.15\*sal)+(0.2\*sal)+(0.08\*sal);

else

sal:=sal+(0.15\*sal)+(0.2\*sal)+(0.08\*sal);

END if;

return sal;

END;

DECLARE

CALC\_SAL NUMBER;

BEGIN

CALC\_SAL:=CostToComp(&staff\_code);

dbms\_output.put\_line(CALC\_SAL);

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.

create or replace procedure staff\_info

as

s\_name staff\_masters.staff\_name%type;

d\_name department\_masters.dept\_name%type;

d\_des designation\_masters.design\_name%type;

s\_sal staff\_masters.staff\_sal%type;

stat VARCHAR2(10);

CURSOR STAFF IS

select s.staff\_name,d.dept\_name,ds.design\_name,s.staff\_sal,

CASE

WHEN S.STAFF\_SAL>(SELECT AVG(STAFF\_SAL) FROM STAFF\_MASTERS) THEN 'GREATER'

WHEN S.STAFF\_SAL<(SELECT AVG(STAFF\_SAL) FROM STAFF\_MASTERS) THEN 'LESSER'

ELSE 'EQUAL'

END

FROM STAFF\_MASTERS S JOIN DEPARTMENT\_MASTERS D ON S.DEPT\_CODE=D.DEPT\_CODE

JOIN DESIGNATION\_MASTERS DS ON S.DESIGN\_CODE=DS.DESIGN\_CODE;

BEGIN

IF NOT STAFF%ISOPEN THEN

OPEN STAFF;

END IF;

LOOP

FETCH STAFF INTO s\_name, d\_name, d\_des, s\_sal, stat;

exit when staff%notfound;

dbms\_output.put\_line(s\_name || ' ' || d\_name || ' ' || d\_des || ' ' || s\_sal || ' ' || stat);

END loop;

close STAFF;

EXCEPTION

WHEN others then

DBMS\_OUTPUT.PUT\_LINE('sorry no records found!!');

END;

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

CREATE TABLE STAFF\_MASTERS\_BACK AS(SELECT \* FROM STAFF\_MASTERS);

TRUNCATE TABLE STAFF\_MASTERS\_BACK;

create or replace procedure staff\_backup(s\_code in staff\_masters.staff\_code%type)

is

exp number;

v\_sal staff\_masters.staff\_sal%type;

BEGIN

insert into staff\_masters\_back(select \* from staff\_masters where staff\_code=s\_code);

select staff\_sal, round(months\_between(sysdate,hiredate)/12)

into v\_sal, exp from staff\_masters where staff\_code=s\_code;

if v\_sal IS NULL THEN

RAISE NO\_DATA\_FOUND;

if exp>5 then

v\_sal := (1.25 \* v\_sal);

END if;

if exp between 2 and 5 then

v\_sal:= (1.20 \* v\_sal);

END if;

update staff\_masters set staff\_sal=v\_sal;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE("NO DATA FOUND! PLS VERIFY");

END;

show error;

execute staff\_backup(100007);

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

create or replace procedure report

as

bcode book\_transactions.book\_code%type:=&enter;

code number;

name1 varchar2(100);

idate book\_transactions.book\_issue\_date%type;

dname designation\_masters.design\_name%type;

rdate book\_transactions.book\_expected\_return\_date%type;

excep exception;

cursor mycur1 is

select s.staff\_code,s.staff\_name,b.book\_issue\_date,d.design\_name,b.book\_expected\_return\_date

from staff\_masters s, book\_transactions b, designation\_masters d

where b.staff\_code=s.staff\_code and s.design\_code=d.design\_code

and s.staff\_code in (select nvl(staff\_code,student\_code) from book\_transactions

where book\_code=bcode and book\_actual\_return\_date is null);

cursor mycur2 is

select s.student\_code,s.student\_name,b.book\_issue\_date,b.book\_expected\_return\_date

from student\_masterss s, book\_transactions b

where b.student\_code=s.student\_code

and s.student\_code in (select nvl(staff\_code,student\_code) from book\_transactions

where book\_code=bcode and book\_actual\_return\_date is null);

BEGIN

IF NOT mycur1%ISOPEN THEN

OPEN mycur1;

END IF;

IF NOT mycur2%ISOPEN THEN

OPEN mycur2;

END IF;

LOOP

FETCH mycur1 into code,name1,idate,dname,rdate;

IF (mycur1%notfound and mycur1%rowcount=0)then

FETCH mycur2 into code,name1,idate,rdate;

IF(mycur2%notfound and mycur2%rowcount=0)then

RAISE excep;

ELSE

EXIT WHEN mycur2%NOTFOUND;

dbms\_output.put\_line(code||' '||name1||' '||idate||' '||dname||' '||rdate);

END IF;

ELSE

EXIT WHEN mycur1%NOTFOUND;

dbms\_output.put\_line(code||' '||name1||' '||idate||' '||dname||' '||rdate);

END IF;

END LOOP;

CLOSE mycur2;

CLOSE mycur1;

EXCEPTION

WHEN excep then

dbms\_output.put\_line('everyone refunded');

END;

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.

CREATE OR REPLACE PACKAGE staffpack

AS

exp NUMBER;

FUNCTION calc\_exp(hdate date) RETURN NUMBER;

PROCEDURE calc\_al(years IN NUMBER,allowance OUT NUMBER);

END staffpack;

CREATE OR REPLACE PACKAGE BODY staffpack AS

FUNCTION calc\_exp(hdate date) RETURN NUMBER IS

BEGIN

exp:=ROUND(MONTHS\_BETWEEN(sysdate,hdate)/12);

RETURN exp;

END calc\_exp;

PROCEDURE calc\_al

(years IN NUMBER,allowance OUT NUMBER) IS

BEGIN

allowance:=years\*3000;

END calc\_al;

END staffpack;

DECLARE

hdate date;

exp\_yrs NUMBER(15);

tot\_al NUMBER(20);

scode staff\_masters.staff\_code%TYPE:=&code;

BEGIN

SELECT hiredate INTO hdate

FROM staff\_masters WHERE staff\_code=scode;

exp\_yrs:=staffpack.calc\_exp(hdate);

staffpack.calc\_al(exp\_yrs,tot\_al);

INSERT INTO staff\_allowance(s\_code, date\_of\_joining,experience, allowance)

VALUES(scode,hdate,exp\_yrs,tot\_al);

END;

/

create table staff\_allowance(s\_code number, date\_of\_joining date,

experience number, allowance number);

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.

create or replace procedure insertrecords

(bcode in book\_transactions.book\_code%type,

code in number)as

code1 number;

exc exception;

doe varchar2(50);

doe1 date;

cursor mycur1 is

select staff\_code from staff\_masters where staff\_code=code;

cursor mycur2 is

select student\_code from student\_masters where student\_code=code;

BEGIN

select to\_char(sysdate+10,'DAY') INTO doe FROM DUAL;

if(doe='SATURDAY') then

select (sysdate+12) INTO doe1 FROM DUAL;

elsif (doe='SUNDAY')then

select (sysdate+11) INTO doe1 FROM DUAL;

else

select (sysdate+10) INTO doe1 FROM DUAL;

END if;

open mycur1;

open mycur2;

loop

fetch mycur1 into code1;

if(mycur1%notfound and mycur1%rowcount=0) then

fetch mycur2 into code1;

if(mycur2%notfound and mycur2%rowcount=0) then

raise exc;

else

exit when(mycur2%notfound);

insert into book\_transactions values(bcode,code1,null,sysdate,doe1,null);

END if;

else

exit when(mycur1%notfound);

insert into book\_transactions values(bcode,null,code1,sysdate,doe1,null);

END if;

END loop;

exception

when exc then

dbms\_output.put\_line('not found');

END;

/

show error;

delete from book\_transactions where book\_issue\_date LIKE '%07-OCT-17%';

execute insertrecords(10000007,100002);

execute insertrecords('10000002','1015');

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

CREATE OR REPLACE FUNCTION count\_Row(t\_name VARCHAR2)

RETURN NUMBER

IS

countrow NUMBER;

BEGIN

EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM '||t\_name INTO countrow;

dbms\_output.put\_line(countrow);

RETURN countrow;

END;

DECLARE

v\_count NUMBER;

BEGIN

v\_count:=count\_Row('staff\_masters');

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.

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

v\_name VARCHAR2(15);

v\_newsal NUMBER;

BEGIN

UPDATE emp\_copy SET sal = sal \* 1.1

WHERE empno = emp\_id;

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

FROM emp\_copy

WHERE empno = emp\_id;

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

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

END;

CODE:

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

v\_name VARCHAR2(15):=' ';

v\_newsal NUMBER:=0;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('EMP\_ID:' || EMP\_ID);

SELECT ename, sal INTO v\_name, v\_newsal FROM employee WHERE empno = emp\_id;

--UPDATE employee SET sal = sal \* 1.1 WHERE empno = emp\_id;

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

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

END;

/

Procedure created.

SQL> exec update\_salary(7369);

PL/SQL procedure successfully completed.

SQL> select sal from employee where empno=7369;

OUTPUT:

PAGE 1

SALARY

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

$1,100

4.13

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

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

BEGIN

DELETE FROM table\_name;

END;

CODE:

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

2 BEGIN

3 EXECUTE IMMEDIATE 'TRUNCATE TABLE table\_name';

4 END;

5 /

Procedure created.

SQL> EXEC GETTABLE(staff\_masters);

PL/SQL procedure successfully completed.

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.

Sample output for deptno 10 is shown below:

Employee Name : CLARK

Employee Job : MANAGER

Employee Salary : 2450

Employee Comission :

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

Employee Name : KING

Employee Job : PRESIDENT

Employee Salary : 5000

Employee Comission :

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

Employee Name : MILLER

Employee Job : CLERK

Employee Salary : 1300

Employee Comission :

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

Number of Employees : 3

Total Salary : 8750

Maximum Salary : 5000

Average Salary : 2916.67

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

Figure 1 :Report

CREATE OR REPLACE PROCEDURE proname25(no dept.deptno%TYPE)

AS

num NUMBER;

CURSOR cname IS SELECT \* FROM employee WHERE deptno=no;

emp employee%rowtype;

sum NUMBER:=0;

avg NUMBER:=0;

nename varchar2(20);

cename NUMBER;

njob varchar2(20);

nsal NUMBER;

ncomm NUMBER;

ndeptno NUMBER;

SALAR NUMBER:=0;

BEGIN

num:=&no;

open cname;

fetch cname into emp;

SELECT ENAME INTO nename FROM employee WHERE deptno=no;

SELECT COUNT(ename) INTO cename FROM employee WHERE deptno=no;

SELECT JOB INTO njob FROM employee WHERE deptno=no;

SELECT COMM INTO ncomm FROM employee WHERE deptno=no;

FOR rec IN cname

LOOP

DBMS\_OUTPUT.PUT\_LINE('DETAILS OF EMPLOYEES:' || 'EMPLOYEE NAME:' || nename || 'NUMBER OF EMPLOYEE:'|| cename ||'EMPLOYEE JOB:' || njob||'EMPLOYEE SALARY:'|| nsal||

'EMPLOYEE COMMISSION:' || ncomm);

END LOOP;

FOR rec IN cname

LOOP

SALAR:=EMP.SAL;

sum:=(sum+SALAR);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('total salary of employee:'|| sum);

FOR rec IN cname

LOOP

avg:=(sum/cename);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('average salary of employee:'|| avg);

FOR rec IN cname

LOOP

select s.nsal from employee s,employee e where s.nsal>e.nsal;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('max salary of employee:'|| s.nsal );

END;

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

SQL> select \*from user\_objects where object\_type in ('PROCEDURE','FUNCTION','PACKAGE');

SQL> select \*from user\_objects where object\_type in ('PROCEDURE','FUNCTION','PACKAGE');

========================================================================================

LAB-4:

Reports

1)

SQL> BREAK ON deptno SKIP PAGE ON empno SKIP 1

SQL> SELECT empno, deptno, ename, sal FROM

2 EMP

3 WHERE sal>800

4 ORDER BY empno,deptno;

Page: 1

EMPNO DEPTNO ENAME SAL

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

7499 30 ALLEN 1600

7521 WARD 1250

Page: 2

EMPNO DEPTNO ENAME SAL

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

7566 20 JONES 2975

Page: 3

EMPNO DEPTNO ENAME SAL

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

7654 30 MARTIN 1250

7698 BLAKE 2850

Page: 4

EMPNO DEPTNO ENAME SAL

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

7782 10 CLARK 2450

Page: 5

EMPNO DEPTNO ENAME SAL

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

7788 20 SCOTT 3000

Page: 6

EMPNO DEPTNO ENAME SAL

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

7839 10 KING 5000

Page: 7

EMPNO DEPTNO ENAME SAL

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

7844 30 TURNER 1500

Page: 8

EMPNO DEPTNO ENAME SAL

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

7876 20 ADAMS 1100

Page: 9

EMPNO DEPTNO ENAME SAL

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

7900 30 JAMES 950

Page:##

EMPNO DEPTNO ENAME SAL

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

7902 20 FORD 3000

Page:##

EMPNO DEPTNO ENAME SAL

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

7934 10 MILLER 1300

13 rows selected.

SQL> SELECT deptno,avg(sal) FROM

2 EMP group by deptno;

Page: 1

DEPTNO AVG(SAL)

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

30 1566.66667

Page: 2

DEPTNO AVG(SAL)

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

20 2175

Page: 3

DEPTNO AVG(SAL)

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

10 2916.66667

Lab 5.

Case Study 1

Goals Implementation of Procedures/Functions ,Packages with Testing and Review

Time 2.5hrs

Consider the following tables for the case study.

Customer\_Masters

Note: Customer type can be either IND or NRI

Account\_Masters Table

Name Null? Type

Account\_Number Not Null Number(6)

Cust\_ID Number(6)

Account\_Type Char(3)

Ledger\_Balance Number(10)

Note: Account type can be either Savings (SAV) or Salary (SAL) account.

For savings account minimum amount should be 5000.

Transaction\_Masters

Name Null? Type

Cust\_Id Not Null Number(6)

Cust\_Name Not Null Varchar2(20)

Address Varchar2(50)

Date\_of\_acc\_creation Date

Customer\_Type Char(3)

Name Null? Type

Transaction\_Id

Not Null Number(6)

Account\_Number Number(6)

Date\_of\_Transaction Date

From\_Account\_Number Not Null Number(6)

To\_Account\_Number Not Null Number(6)

Amount Not Null Number(10)

Transaction\_Type Not Null Char(2)

Note: Transaction type can be either Credit (CR) or Debit (DB).

Procedure and function should be written inside a package.

All validations should be taken care.

5.1 Create appropriate Test Cases for the case study followed up by Self/Peer to Peer

Review and close any defects for the same.

5.2Write a procedure to accept customer name, address, and customer type and account

type. Insert the details into the respective tables.

5.3. Write a procedure to accept customer id, amount and the account number to which

the customer requires to transfer money. Following validations need to be done

Customer id should be valid

From account number should belong to that customer

To account number cannot be null but can be an account which need not exist in

account masters (some other account)

Adequate balance needs to be available for debit

5.4 Ensure all the Test cases defined are executed. Have appropriate Self/Peer to Peer

Code Review and close any defects for the same.

Lab 6. Case Study 2

Goals Implementation of Procedures/Functions ,Packages with Testing and Review

Time 2.5hrs

Consider the following table (myEmp) structure for the case study

EmpNo Ename City Designation Salary

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

The following procedure accepts Task number and based on the same performs an

appropriate task.

PROCEDURE run\_task (task\_number\_in IN INTEGER)

IS

BEGIN

IF task\_number\_in = 1

THEN

add\_emp;

--should add new emps in myEmp.

--EmpNo should be inserted through Sequence.

--All other data to be taken as parameters.Default location is Mumbai.

END IF;

IF task\_number\_in = 2

THEN

raise\_sal;

--should modify salary of an existing emp.

--should take new salary and empno as input parameters

--Should handle exception in case empno not found

--upper limit of rasing salary is 30%. should raise exception appropriately

END IF;

IF task\_number\_in = 3

THEN

remove\_emp;

--should remove an existing emp

--should take empno as parameter

--Handle exception if empno not available

END IF;

END run\_task;

Example 7: Sample Oracle Procedure

However ,it has been observed the method adopted in above procedure is inefficient.

6.1

Create appropriate Test Cases for the case study followed up by Self/Peer to Peer

Review and close any defects for the same.

6.2

Recreate the procedure (run\_task) which is more efficient in performing the same.

6.3

Also, create relevant procedures (add\_emp , raise\_sal ,remove\_emp)

with relevant logic (read comments)to verify the same.

6.4 ExtEND the above implementation using Packages

6.5) Ensure all the Test cases defined are executed. Have appropriate Self/Peer to Peer

Code Review and close any defects for the same.