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.**

**Input:**

DECLARE

CURSOR C\_info IS

SELECT max(Staff\_Sal) FROM Staff\_Master WHERE Dept\_Code = &Dept\_Code;

Max\_sal Staff\_Master.Staff\_Sal%TYPE;

BEGIN

OPEN C\_Info;

FETCH C\_Info INTO Max\_Sal;

IF C\_Info%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Maximum Salary is:'||Max\_Sal);

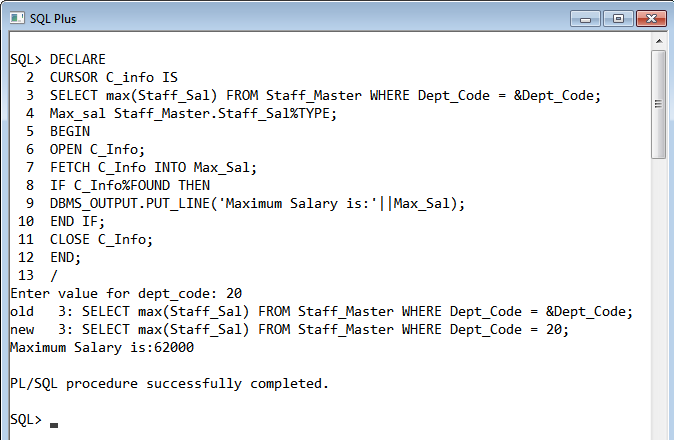
END IF;

CLOSE C\_Info;

END;

/

**OUTPUT:**



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

**years.**

**INPUT:**

CREATE OR REPLACE FUNCTION Age(Birthdate in Date)

RETURN NUMBER IS

age\_in\_years Number(20);

BEGIN

age\_in\_years:=ROUND(months\_between(SYSDATE,Birthdate)/12);

RETURN age\_in\_years;

EXCEPTION

WHEN OTHERS THEN

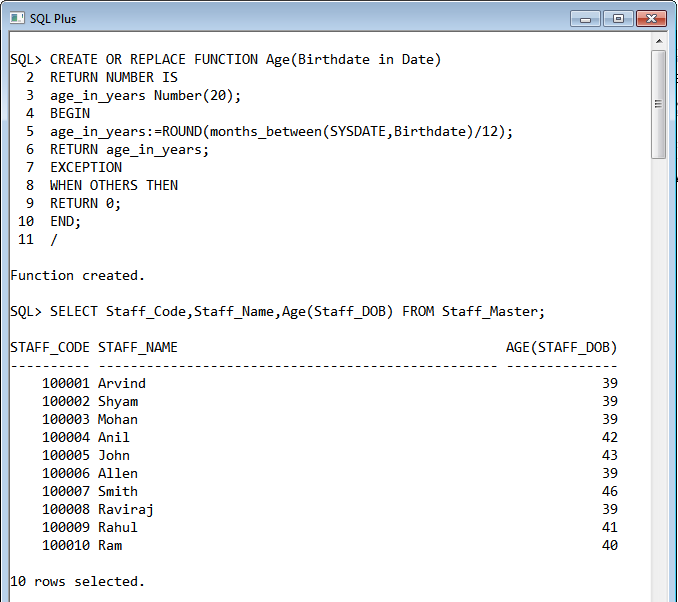
RETURN 0;

END;

/

SELECT Staff\_Code,Staff\_Name,Age(Staff\_DOB) FROM Staff\_Master;

**OUTPUT:**



**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.**

**INPUT:**

CREATE OR REPLACE PROCEDURE To\_Upper(Staff\_ID in NUMBER) IS

CURSOR C\_Name IS

SELECT Staff\_Code,Staff\_Name FROM Staff\_Master WHERE Staff\_Code=Staff\_ID;

var C\_Name%ROWTYPE;

Name\_Null EXCEPTION;

BEGIN

OPEN C\_Name;

FETCH C\_Name INTO var;

IF C\_Name%FOUND THEN

IF var.Staff\_Name IS NULL THEN

RAISE Name\_Null;

ELSE

UPDATE Staff\_Master SET Staff\_Name=UPPER(var.Staff\_Name) WHERE Staff\_Code=var.Staff\_Code;

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('No Record Found');

END IF;

COMMIT;

CLOSE C\_Name;

EXCEPTION

WHEN Name\_Null THEN

DBMS\_OUTPUT.PUT\_LINE('Name not found for the given Staff Name');

END;

/

BEGIN

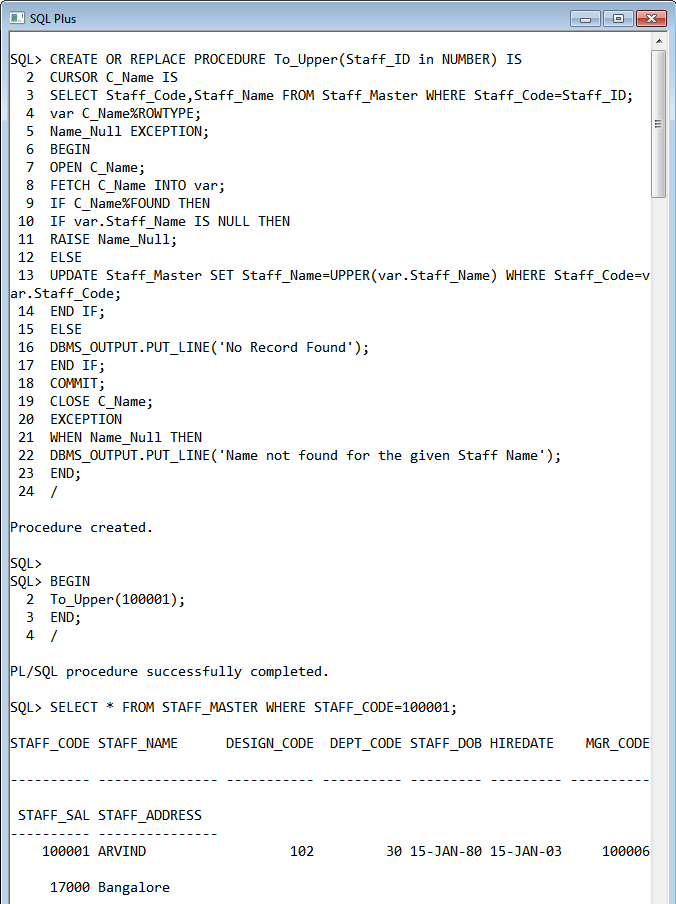
To\_Upper(100001);

END;

/

SELECT \* FROM STAFF\_MASTER WHERE STAFF\_CODE=100001;

**OUTPUT:**



**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.**

**INPUT:**

CREATE OR REPLACE PROCEDURE Find\_Manager(Staff\_ID in NUMBER) IS

CURSOR C\_Name IS

SELECT s1.Staff\_Code,s1.Staff\_Name,s1.Dept\_Code,s2.Staff\_Name"Manager\_Name"

FROM Staff\_Master s1,Staff\_Master s2 WHERE s2.Staff\_Code=s1.MGR\_Code and s1.Staff\_Code=Staff\_ID;

Code staff\_Master.Staff\_Code%TYPE;

Name Staff\_Master.Staff\_Name%TYPE;

Dept Staff\_Master.Dept\_Code%TYPE;

Manager Staff\_Master.Staff\_Name%TYPE;

BEGIN

OPEN C\_Name;

DBMS\_OUTPUT.PUT\_LINE('Staff Code '||' Staff Name '||' Dept Code '||' Manager Name');

LOOP

FETCH C\_Name INTO Code,Name,Dept,Manager;

EXIT WHEN C\_Name%NOTFOUND;

IF C\_Name%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(Code||' '||Name||' '||Code||' '||Manager);

END IF;

END LOOP;

COMMIT;

CLOSE C\_Name;

END;

/

BEGIN

Find\_Manager(100008);

END;

/

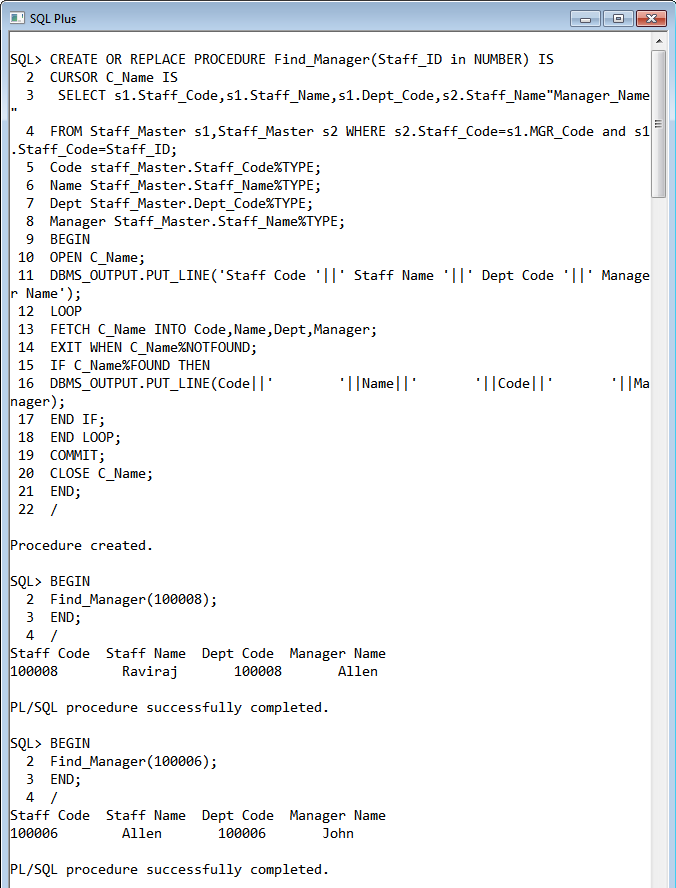
BEGIN

Find\_Manager(100006);

END;

/

**OUTPUT:**



**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

**INPUT:**

CREATE OR REPLACE FUNCTION CTC(Staff\_ID in NUMBER)

RETURN number is

sal staff\_master.staff\_sal%type;

DOJ staff\_master.hiredate%type;

ctc number(10);

DA number(10);

HRA number(10);

TA number(10);

Spl\_Allow number(10);

exp number(10);

BEGIN

select staff\_sal,hiredate INTO sal,DOJ FROM staff\_master WHERE staff\_code=Staff\_ID;

DA:=sal\*15/100;

HRA:=sal\*20/100;

TA:=sal\*8/100;

exp:=round((sysdate-DOJ)/365);

IF exp>=1 and exp<2 then

Spl\_Allow:=sal\*10/100;

ELSIF exp>=2 and exp<3 then

Spl\_Allow:=sal\*20/100;

ELSIF exp>4 then

Spl\_Allow:=sal\*30/100;

ELSE

Spl\_Allow:=0;

END IF;

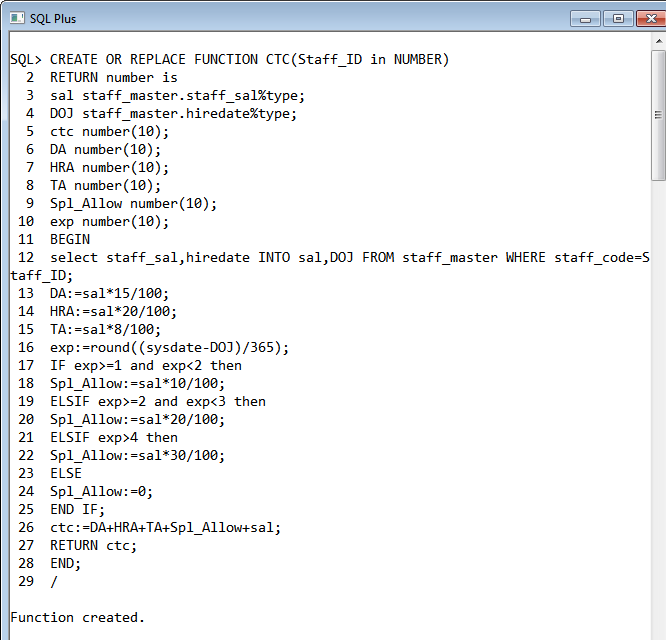
ctc:=DA+HRA+TA+Spl\_Allow+sal;

RETURN ctc;

END;

/

**OUTPUT:**



**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.

**INPUT:**

CREATE OR REPLACE PROCEDURE Staff\_Info IS

CURSOR C\_Staff IS

SELECT s.Staff\_Name,s.Dept\_Code,d1.Dept\_Name,d2.Design\_Name,s.Staff\_Sal

FROM Staff\_Master s,Department\_Master d1,Designation\_Master d2

WHERE s.Dept\_Code=d1.Dept\_Code and s.Design\_Code=d2.Design\_Code;

var C\_Staff%ROWTYPE;

Avrg Staff\_Master.Staff\_Sal%TYPE;

Status VARCHAR2(20);

counter NUMBER:=0;

BEGIN

OPEN C\_Staff;

DBMS\_OUTPUT.PUT\_LINE('Staff Name '||' Department Name '||' Designation '||' Salary '||' Status');

DBMS\_OUTPUT.PUT\_LINE('------------------------------------------------------------------------------------------');

LOOP

FETCH C\_Staff INTO var;

EXIT WHEN C\_Staff%NOTFOUND;

IF C\_Staff%FOUND THEN

counter:=counter+1;

SELECT Avg(Staff\_Sal) INTO Avrg FROM Staff\_Master WHERE Dept\_Code=var.Dept\_Code;

IF var.Staff\_Sal > Avrg THEN

Status:='Greater';

ELSIF var.Staff\_Sal < Avrg THEN

Status:='Lesser';

ELSE

Status:='Equal';

END IF;

DBMS\_OUTPUT.PUT\_LINE(var.Staff\_Name||' '||var.Dept\_Name||' '||var.Design\_Name||' '||var.Staff\_Sal||' '||Status);

END IF;

END LOOP;

IF counter=0 THEN

DBMS\_OUTPUT.PUT\_LINE('There is no matching Record.');

END IF;

COMMIT;

CLOSE C\_Staff;

END;

/

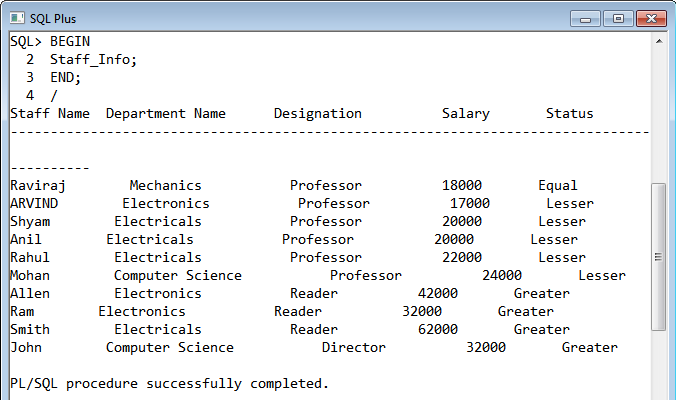
BEGIN

Staff\_Info;

END;

/

**OUTPUT:**



**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

**INPUT:**

CREATE OR REPLACE PROCEDURE Update\_Salary(Staff\_ID NUMBER) IS

Exp NUMBER;

Increment NUMBER;

Salary NUMBER;

BEGIN

SELECT round(months\_between(SYSDATE,HIREDATE)/12),Staff\_Sal INTO Exp,Salary FROM Staff\_Master WHERE STaff\_Code=Staff\_ID;

IF Exp < 2 THEN

Increment:=0;

ELSIF (Exp > 2 AND Exp < 5) THEN

Increment:=0.20\*Salary;

ELSE

Increment:=0.25\*Salary;

END IF;

INSERT INTO Staff\_Master\_Back

SELECT \* FROM Staff\_Master WHERE Staff\_Code=Staff\_ID;

UPDATE Staff\_Master

SET STaff\_Sal=Staff\_Sal+Increment

WHERE Staff\_Code = Staff\_ID;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No Record Found');

END;

/

BEGIN

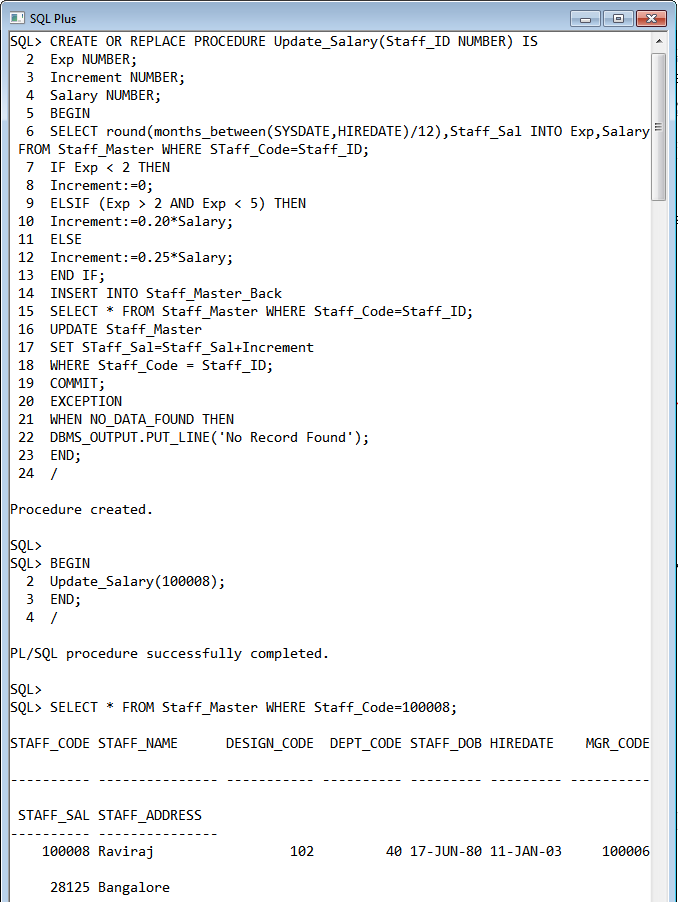
Update\_Salary(100008);

END;

/

SELECT \* FROM Staff\_Master WHERE Staff\_Code=100008;

**OUTPUT:**



**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**

**INPUT:**

CREATE OR REPLACE PROCEDURE Pending\_Books(Book\_ID NUMBER) IS

CURSOR C\_Stu\_Books IS

SELECT s1.Student\_Code,s1.Student\_Name,b.Book\_Issue\_Date,b.Book\_Expected\_Return\_Date

FROM Student\_Master s1,Book\_Transactions b

WHERE s1.Student\_Code=b.Student\_Code AND b.Book\_Code=Book\_ID;

var1 C\_Stu\_Books%ROWTYPE;

CURSOR C\_Sta\_Books IS

SELECT s2.Staff\_Code,s2.Staff\_Name,d.Design\_Name,b.Book\_Issue\_Date,b.Book\_Expected\_Return\_Date

FROM Staff\_Master s2,Designation\_master d,Book\_Transactions b

WHERE s2.Staff\_Code=b.Staff\_Code AND s2.Design\_code=d.Design\_Code AND b.Book\_Code=Book\_ID;

var2 C\_Sta\_Books%ROWTYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Student/Staff Code'||' '||'Student/Staff Name'||' '||'Issue Date'||' '||'Designation'||' '||'Expected Ret\_Date');

OPEN C\_Stu\_Books;

LOOP

FETCH C\_Stu\_Books INTO var1;

EXIT WHEN C\_Stu\_Books%NOTFOUND;

IF C\_Stu\_Books%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(var1.Student\_Code||' '||var1.Student\_Name||' '||var1.Book\_Issue\_Date||' '||'NA'||' '||var1.Book\_Expected\_Return\_date);

END IF;

END LOOP;

CLOSE C\_Stu\_Books;

OPEN C\_Sta\_Books;

LOOP

FETCH C\_Sta\_Books INTO var2;

EXIT WHEN C\_Sta\_Books%NOTFOUND;

IF C\_Sta\_Books%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(var2.Staff\_Code||' '||var2.Staff\_Name||' '||var2.Book\_Issue\_Date||' '||var2.Design\_Name||' '||var2.Book\_Expected\_Return\_date);

END IF;

END LOOP;

CLOSE C\_Sta\_Books;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No Record Found');

END;

/

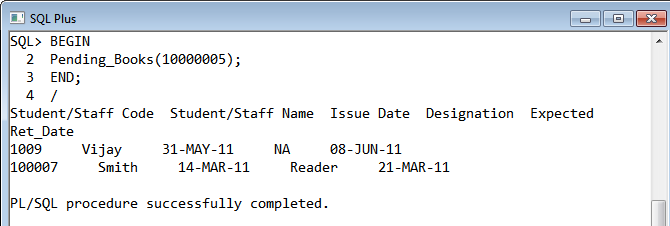
BEGIN

Pending\_Books(10000005);

END;

/

**OUTPUT:**



**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.**

**INPUT:**

CREATE TABLE STAFF\_ALLOWANCE (

STAFF\_CODE NUMBER(20),

HIREDATE DATE,

EXPR NUMBER(10),

ALLOWANCE NUMBER(10,2));

CREATE OR REPLACE PACKAGE staff\_package

AS

FUNCTION EXPERIENCE(V\_HIREDATE IN STAFF\_MASTER.HIREDATE%TYPE) RETURN NUMBER;

PROCEDURE ALLOWANCE(S\_CODE IN NUMBER);

END;

/

CREATE OR REPLACE PACKAGE BODY staff\_package

AS

FUNCTION EXPERIENCE(V\_HIREDATE IN STAFF\_MASTER.HIREDATE%TYPE)

RETURN NUMBER

IS

c\_expr NUMBER;

BEGIN

c\_expr := ROUND(MONTHS\_BETWEEN(SYSDATE,V\_HIREDATE)/12,0);

RETURN c\_expr;

END EXPERIENCE;

PROCEDURE ALLOWANCE(S\_CODE IN NUMBER)

IS

C\_EXPR NUMBER;

ADDITIONAL\_ALLOWANCE NUMBER;

V\_HIREDATE STAFF\_MASTER.HIREDATE%TYPE;

BEGIN

SELECT HIREDATE

INTO V\_HIREDATE

FROM STAFF\_MASTER

WHERE STAFF\_CODE = S\_CODE;

C\_EXPR := staff\_package.EXPERIENCE(V\_HIREDATE);

DBMS\_OUTPUT.PUT\_LINE(C\_EXPR);

ADDITIONAL\_ALLOWANCE := C\_EXPR \* 3000;

INSERT INTO STAFF\_ALLOWANCE VALUES(S\_CODE, V\_HIREDATE, C\_EXPR, ADDITIONAL\_ALLOWANCE);

DBMS\_OUTPUT.PUT\_LINE(ADDITIONAL\_ALLOWANCE);

END ALLOWANCE;

END staff\_package;

/

DECLARE

S\_CODE STAFF\_MASTER.STAFF\_CODE%TYPE;

BEGIN

S\_CODE:=&staff\_code;

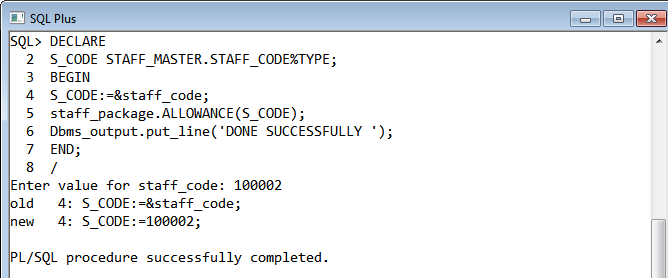
staff\_package.ALLOWANCE(S\_CODE);

Dbms\_output.put\_line('DONE SUCCESSFULLY ');

END;

/

**OUTPUT:**



**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.**

**INPUT:**

CREATE OR REPLACE PROCEDURE Issue\_Books(Book\_ID NUMBER,Stu\_Sta\_ID NUMBER) IS

Issue\_Date DATE;

Expect\_Return\_date DATE;

BEGIN

Issue\_Date:=SYSDATE;

IF TO\_CHAR(Issue\_Date+10,'dy')='sat' THEN

Expect\_Return\_Date:=Issue\_Date+12;

ELsIF TO\_CHAR(Issue\_Date+10,'dy')='sun' THEN

Expect\_Return\_Date:=Issue\_Date+11;

ELSE

Expect\_Return\_Date:=Issue\_Date+10;

END IF;

IF LENGTH(TO\_CHAR(Stu\_Sta\_ID))=4 THEN

INSERT INTO Book\_Transactions(Book\_Code,Student\_Code,Book\_Issue\_Date,Book\_Expected\_Return\_Date)

VALUES(Book\_ID,Stu\_Sta\_ID,Issue\_Date,Expect\_Return\_Date);

ELSIF LENGTH(TO\_CHAR(Stu\_Sta\_ID))=6 THEN

INSERT INTO Book\_Transactions(Book\_Code,Staff\_Code,Book\_Issue\_Date,Book\_Expected\_Return\_Date)

VALUES(Book\_ID,Stu\_Sta\_ID,Issue\_Date,Expect\_Return\_Date);

ELSE

DBMS\_OUTPUT.PUT\_LINE('INVALID Student or Staff ID');

END IF;

COMMIT;

END;

/

BEGIN

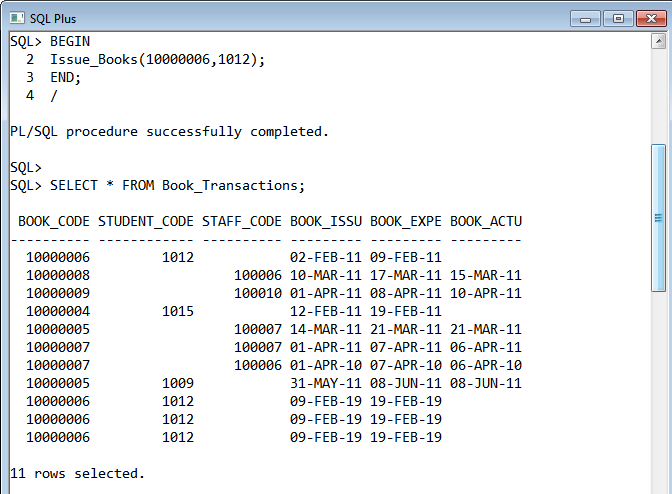
Issue\_Books(10000006,1012);

END;

/

SELECT \* FROM Book\_Transactions;

**OUTPUT:**



**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.**

**INPUT:**

CREATE or REPLACE FUNCTION Get\_Total\_Records(Table\_Name IN varchar2)

RETURN NUMBER IS

Total\_Records NUMBER;

BEGIN

EXECUTE IMMEDIATE 'SELECT count(\*) FROM ' || Table\_Name INTO Total\_Records;

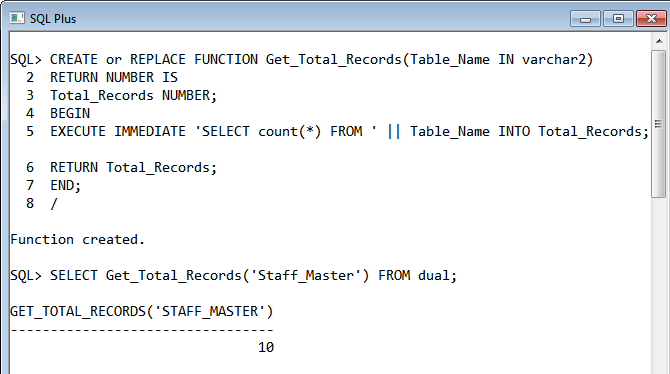
RETURN Total\_Records;

END;

/

SELECT Get\_Total\_Records('Staff\_Master') FROM dual;

**OUTPUT:**



**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;

**INPUT:**

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);

EXCEPTION

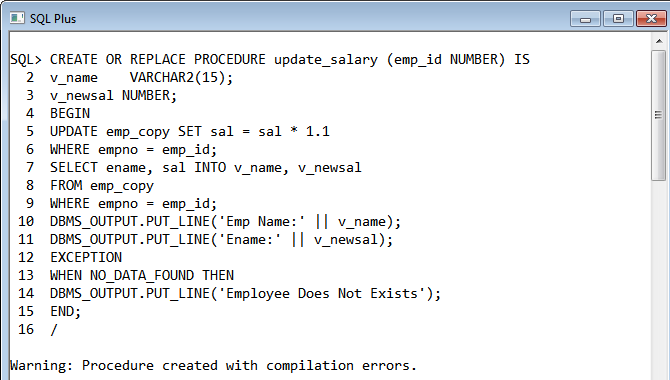
WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Employee Does Not Exists');

END;

/

**OUTPUT:**



**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;

**INPUT:**

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

BEGIN

EXECUTE IMMEDIATE 'DELETE FROM ' || table\_name;

END;

/

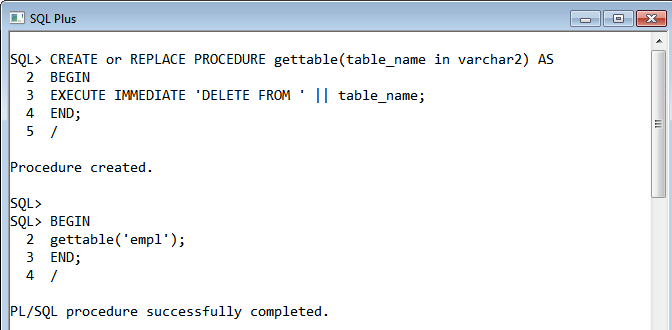
BEGIN

gettable('empl');

END;

/

**OUTPUT:**



**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**

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

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

**INPUT:**

CREATE or REPLACE PROCEDURE Combined\_Report(Total\_Emp IN NUMBER,TOtal\_Sal IN NUMBER,Max\_Sal IN NUMBER,Avg\_Sal IN NUMBER) AS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Number of Employees:'||Total\_Emp);

DBMS\_OUTPUT.PUT\_LINE('Total Salary:'||Total\_Emp);

DBMS\_OUTPUT.PUT\_LINE('Maximum Salary:'||Max\_Sal);

DBMS\_OUTPUT.PUT\_LINE('Average Salary:'||TRUNC(Avg\_Sal,2));

DBMS\_OUTPUT.PUT\_LINE('---------------------------');

END;

/

CREATE or REPLACE PROCEDURE Report(Dept\_Code in NUMBER) AS

CURSOR Emp\_Details IS

SELECT EName,Job,Sal,Comm FROM Emp WHERE DeptNo=Dept\_Code;

var Emp\_Details%ROWTYPE;

Ecount NUMBER:=0;

Total\_Sal NUMBER:=0;

Max\_Sal NUMBER:=0;

Avg\_Sal NUMBER:=0;

BEGIN

OPEN Emp\_Details;

LOOP

FETCH Emp\_Details INTO var;

EXIT WHEN Emp\_Details%NOTFOUND;

IF Emp\_Details%FOUND THEN

Ecount:=Ecount+1;

Total\_Sal:=Total\_Sal+var.Sal;

IF Max\_Sal < var.Sal THEN

Max\_Sal:=var.Sal;

END IF;

Avg\_Sal:=Total\_Sal/Ecount;

DBMS\_OUTPUT.PUT\_LINE('Employee Name:'||var.EName);

DBMS\_OUTPUT.PUT\_LINE('Employee Job:'||var.Job);

DBMS\_OUTPUT.PUT\_LINE('Employee Salary:'||var.Sal);

DBMS\_OUTPUT.PUT\_LINE('Employee Commission:'||var.Comm);

DBMS\_OUTPUT.PUT\_LINE('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*');

END IF;

END LOOP;

CLOSE Emp\_Details;

Combined\_Report(Ecount,Total\_Sal,Max\_Sal,Avg\_Sal);

END;

/

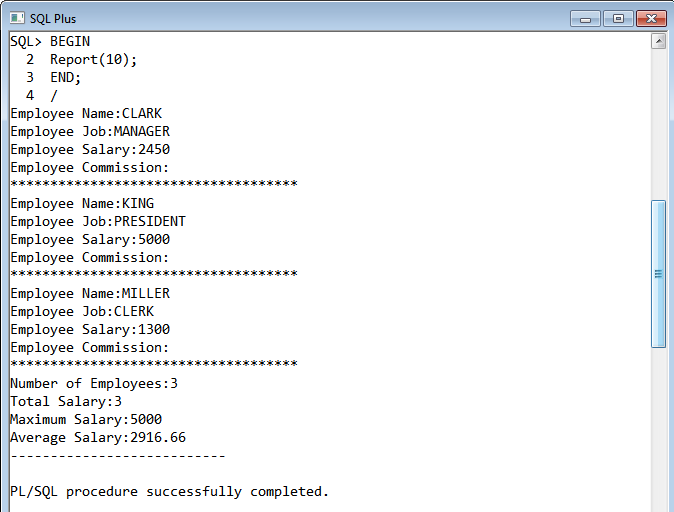
BEGIN

Report(10);

END;

/

**OUTPUT:**



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

**Data Dictionary.**

**INPUT:**

SELECT OBJECT\_NAME,OBJECT\_TYPE FROM USER\_OBJECTS WHERE OBJECT\_TYPE IN ('FUNCTION','PROCEDURE','PACKAGE');

**OUTPUT:**

