**PL/SQL**

**Day-1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Simple\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

SQL> set serveroutput on;

SQL> DECLARE

v\_comp\_name varchar2(20);

BEGIN

v\_comp\_name:='iGate Corp Ltd ';

DBMS\_OUTPUT.PUT\_LINE(v\_comp\_name || 'XYZ');

END;

/

iGate Corp Ltd XYZ

PL/SQL procedure successfully completed.

**Demo-1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RowType \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

n\_Record staff\_masters%rowtype;

BEGIN

SELECT \* into n\_Record FROM staff\_masters WHERE staff\_code=100001;

UPDATE staff\_masters SET staff\_sal=staff\_sal+101 WHERE staff\_code=100001;

END;

**Demo-2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RowType\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

r\_dept\_info department\_masters%ROWTYPE;

BEGIN

r\_dept\_info.dept\_code:=70;

r\_dept\_info.dept\_name:='Personal';

INSERT INTO department\_masters VALUES r\_dept\_info ;

END;

PL/SQL procedure successfully completed.

**Demo-2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Sub Types\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

set serveroutput on;

DECLARE

SUBTYPE HIREDATE IS DATE NOT NULL;

PFDATE HIREDATE := SYSDATE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(PFDATE);

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Record Type\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

TYPE costRec is RECORD

(customer\_id number,cust\_name varchar2(20));

v\_cust\_rec costRec;

BEGIN

v\_cust\_rec.customer\_id:=20;

v\_cust\_rec.cust\_name:='Smith';

DBMS\_OUTPUT.PUT\_LINE(v\_cust\_rec.customer\_id ||' '|| v\_cust\_rec.cust\_name);

END;

/

0 Smith

L/SQL procedure successfully completed.

**Demo-1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Table Data Type\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

SQL> DECLARE

TYPE staff\_table is table of

staff\_masters.staff\_name%type INDEX BY BINARY\_INTEGER;

staff\_tab staff\_table;

BEGIN

staff\_tab(1):='Smith';

UPDATE staff\_masters SET staff\_sal=1.1\*staff\_sal WHERE staff\_name=staff\_tab(1);

END;

PL/SQL procedure successfully completed.

**Demo-2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Table Data Type\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

TYPE table\_staff\_name IS TABLE OF

staff\_masters.staff\_name%TYPE

INDEX BY BINARY\_INTEGER;

v\_staff\_names table\_staff\_name;

v\_sn staff\_masters.staff\_name%TYPE;

v\_sc staff\_masters.staff\_code%TYPE;

BEGIN

v\_staff\_names(1):='aaa';

v\_staff\_names(2):='bbb';

v\_staff\_names(3):='ccc';

FOR i IN 1..3

LOOP

DBMS\_OUTPUT.PUT\_LINE('i = '||i||' '|| v\_staff\_names(i));

END LOOP;

FOR j IN 1..3

LOOP

UPDATE staff\_masters SET staff\_name=v\_staff\_names(j)

WHERE staff\_code=&v\_sc;

END LOOP;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IF ELSE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

D VARCHAR2(3):=TO\_CHAR(SYSDATE,'DY');

BEGIN

IF D='SAT' THEN

DBMS\_OUTPUT.PUT\_LINE('Enjoy UR WeekEnd');

ELSIF D='SUN' THEN

DBMS\_OUTPUT.PUT\_LINE('Enjoy Ur Sunday');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Have a NICE day');

END IF;

END;/

Enjoy Ur Sunday

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LOOP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_counter number :=60;

BEGIN

LOOP

INSERT INTO department\_masters VALUEs(v\_counter,'newdept');

v\_counter:=v\_counter+1;

END LOOP;

COMMIT;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Loop Exit When\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_counter number :=61;

BEGIN

LOOP

INSERT INTO department\_masters VALUES (v\_counter,'NEWDEPT'||v\_counter);

DELETE FROM emp WHERE deptno=v\_counter;

v\_counter:=v\_counter+1;

EXIT WHEN v\_counter>99;

END LOOP;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.FOR LOOP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_counter number:=41;

BEGIN

FOR loop\_counter IN 2..5

LOOP

INSERT INTO dept(deptno,dname)

VALUES (v\_counter,'NEW DEPT'||v\_counter);

v\_counter:=v\_counter+1;

END LOOP;

COMMIT;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IMPLICIT CURSUR\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

BEGIN

UPDATE dept SET dname='Production' WHERE deptno=90;

IF SQL%NOTFOUND THEN

INSERT INTO department\_masters VALUES(90,'Production');

END IF;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IMPLICIT CURSUR \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

BEGIN

UPDATE dept SET dname='Account' WHERE deptno=40;

IF SQL%ROWCOUNT=0 THEN

INSERT INTO department\_masters VALUES(40,'Account');

END IF;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Implicit CURSORE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

BEGIN

UPDATE dept SET dname='Account'

WHERE deptno=40;

IF SQL%ROWCOUNT=0 THEN

INSERT INTO department\_masters VALUES(40,'Account');

END IF;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Explicit CURSORE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_salary emp.sal%TYPE;

v\_empno emp.empno%TYPE;

CURSOR c\_empsal IS

SELECT empno,sal FROM emp WHERE sal>2500;

BEGIN

IF NOT c\_empsal%ISOPEN THEN

OPEN c\_empsal;

END IF;

LOOP

FETCH c\_empsal INTO v\_salary,v\_empno;

EXIT WHEN c\_empsal%NOTFOUND;

UPDATE emp SET sal=1.1\*v\_salary WHERE empno=v\_empno;

END LOOP;

CLOSE c\_empsal;

COMMIT;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* Explicit CURSORE-Normal Loop\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_staff\_code staff\_masters.staff\_code%TYPE;

v\_staff\_name staff\_masters.staff\_name%TYPE;

CURSOR cur\_staff\_sal IS

SELECT staff\_code,staff\_name

FROM staff\_masters

WHERE staff\_sal>30000;

BEGIN

IF NOT cur\_staff\_sal%ISOPEN THEN

OPEN cur\_staff\_sal;

END IF;

LOOP

FETCH cur\_staff\_sal INTO v\_staff\_code,v\_staff\_name;

EXIT WHEN cur\_staff\_sal %NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(' Code :'||v\_staff\_code||

' Name : '||v\_staff\_name);

END LOOP;

CLOSE cur\_staff\_sal;

END;

/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Explicit CURSORE** -While\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DECLARE

v\_sno student\_marks.student\_code%type;

CURSOR c\_stud\_marks is SELECT student\_code from student\_marks;

BEGIN

OPEN c\_stud\_marks;

FETCH c\_stud\_marks INTO v\_sno ;

WHILE c\_stud\_marks%FOUND

LOOP

UPDATE student\_marks SET subject3=subject3+10 WHERE student\_code=v\_sno;

FETCH c\_stud\_marks INTO v\_sno ;

END LOOP;

CLOSE c\_stud\_marks;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Explicit Cursor-For-Loop\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

cursor c\_stud\_marks IS SELECT \* FROM student\_marks;

total\_marks number(4);

BEGIN

FOR mks IN c\_stud\_marks

LOOP

total\_marks:= mks.subject1+mks.subject2+mks.subject3;

IF (total\_marks>220) THEN

INSERT into performance VALUES(mks.student\_code,total\_marks);

END IF;

END LOOP;

END;

**\*\*\*\*\*\*\*\*UPDATE FOR OF Where Current Off \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**/\***

Processing Explicit Cursors: Using FOR UPDATE

* The FOR UPDATE clause is part of the SELECT statement of the cursor. It is the last clause of the SELECT statement after the ORDER BY clause (if present).
* Normally, a SELECT operation does not lock the rows being accessed. This allows others to change the data selected by the SELECT statement. Besides, at OPEN time the active set consists of changes which were committed. Any changes made after OPEN even if they are committed, are not reflected in the active result set unless the cursor is reopened. This results in Data Inconsistency.
  + If the FOR UPDATE clause is present, exclusive “row locks” are taken on the rows in the active set.
  + These locks prevent other sessions / users from changing these rows unless the changes are committed.
* If another session / user already has locks on the rows in the active set, then the SELECT FOR UPDATE will wait indefinitely for these locks to be released. This statement will hang the system till the locks are released.
  + To avoid this NOWAIT clause is used.
  + With the NOWAIT clause SELECT FOR UPDATE will not wait for the locks acquired by previous sessions to be released and will return immediately with an ORACLE error**.**

Processing Cursors: Using WHERE CURRENT OF:

Note:

* When querying multiple tables you can use the FOR UPDATE OF column to confine row locking for a particular table.

Processing Cursors: Using WHERE CURRENT OF (contd.):

Note:

* If you are using NOWAIT clause, then OF Column\_List is essential for syntax purpose.
* Any COMMIT should be done after the processing is over in a CURSOR LOOP which has FOR UPDATE clause. This is because after commit locks will be released, the cursor will become invalid, and further FETCH will result in an error.
* This problem can be solved by using primary key. Using primary key simulates the WHERE CURRENT of clause, however it does not create any locks.

**\*/**

**\*\*\*\*\*\*\*\*\*\*\*\*\*Cursor Select For Update Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

CURSOR c\_staff IS select staff\_code,

staff\_masters.design\_code FROM staff\_masters,

designation\_masters WHERE

design\_name='Professor' and staff\_sal>20000

and

staff\_masters.design\_code=designation\_masters.design\_code

FOR UPDATE OF staff\_masters.design\_code NOWAIT;

d\_code designation\_masters.design\_code%type;

BEGIN

SELECT design\_code INTO

d\_code FROM designation\_masters

WHERE design\_name='Director';

FOR v\_rec IN c\_staff

LOOP

UPDATE staff\_masters SET design\_code =d\_code WHERE current of c\_staff;

END LOOP;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parameter Cursor\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

CURSOR cur\_staff\_dept\_det(deptno NUMBER) IS

SELECT staff.staff\_code,staff.staff\_name,

staff.dept\_code,staff.staff\_sal,

dept.dept\_name FROM

staff\_masters staff,department\_masters dept

WHERE staff.dept\_code=dept.dept\_code AND

staff.dept\_code=deptno;

rec\_staff\_details cur\_staff\_dept\_det%ROWTYPE;

BEGIN

OPEN cur\_staff\_dept\_det(&deptno);

LOOP

FETCH cur\_staff\_dept\_det INTO rec\_staff\_details;

EXIT WHEN cur\_staff\_dept\_det%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(

' Code :'||rec\_staff\_details.staff\_code||

' Name : '|| rec\_staff\_details.staff\_name||

' Dept Code:' ||rec\_staff\_details.dept\_code||

' Salary : '||rec\_staff\_details.staff\_sal||

' Dept Name :'||rec\_staff\_details.dept\_name);

END LOOP;

CLOSE cur\_staff\_dept\_det;

END;

/

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Parameter Cursor Using For Loop\*\*\*\*\*\*\*\*\*\*\***

DECLARE

CURSOR cur\_staff\_details(deptno NUMBER)

IS SELECT \* FROM staff\_masters

WHERE dept\_code=deptno;

v\_dno NUMBER;

BEGIN

v\_dno:=&num;

FOR v\_staff\_details\_rec IN cur\_staff\_details(v\_dno)

LOOP

DBMS\_OUTPUT.PUT\_LINE(' Code :'||v\_staff\_details\_rec.staff\_c

ode ||

' Name :'||v\_staff\_details\_rec.staff\_n

ame ||

'Dept No: '||v\_staff\_details\_rec.dept\_

code);

END LOOP;

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ref Cursor Normal Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

TYPE staffcurtype IS REF CURSOR RETURN staff\_masters%rowtype;

staff\_cv staffcurtype;

staff\_cur staff\_masters%ROWTYPE;

BEGIN

open staff\_cv FOR SELECT \* FROM staff\_masters;

LOOP

EXIT WHEN staff\_cv%NOTFOUND;

FETCH staff\_cv INTO staff\_cur;

INSERT into temp\_table VALUES(staff\_cur.staff\_code,staff\_cur.staff\_name,staff\_cur.staff\_sal);

END LOOP;

CLOSE staff\_cv;

END;

**\*\*\*\*\*\*\*\*\*Ref Cursor With Function-Package Normal \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

SQL> CREATE OR REPLACE PACKAGE cv\_types AS

TYPE DeptCurTyp is REF CURSOR RETURN dept%ROWTYPE;

END

cv\_types;

Package created.

CREATE OR REPLACE PROCEDURE dept\_rpt

(dept\_cv IN OUT cv\_types.DeptCurTyp) AS

BEGIN

OPEN dept\_cv FOR SELECT \* FROM dept;

END;

SQL> VARIABLE odcv REFCURSOR

SQL> VARIABLE odcv REFCURSOR;

SQL> EXECUTE dept\_rpt(:odcv);

PL/SQL procedure successfully completed.

SQL> PRINT odcv;

**DEPTNO DNAME LOC**

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

10 ACCOUNTING NEW YORK

20 RESEARCH DALLAS

30 SALES CHICAGO

40 OPERATIONS BOSTON

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ref-Cursor-Functiom\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE FUNCTION dept\_fn RETURN cv\_types.DeptCurTyp IS

resultset cv\_types.DeptCurTyp;

BEGIN

OPEN resultset FOR SELECT \* FROM DEPT;

RETURN(resultset);

END;

VARIABLE rc REFCURSOR

EXECUTE :rc := dept\_fn

PRINT rc

**\*\*\*\*\*\*\*\*\*REF Cursor –Strong Ref Cursor Type\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE PROCEDURE PROC1

IS

TYPE staffcurtype IS REF CURSOR RETURN staff\_masters%ROWTYPE;

staff\_cv staffcurtype; --Declare cursor variable

PROCEDURE Proc2(p\_staff\_cv staffcurtype)

AS

staff\_cur staff\_masters%rowtype;

BEGIN

LOOP

EXIT WHEN staff\_cv%NOTFOUND;

FETCH staff\_cv INTO staff\_cur;

DBMS\_OUTPUT.PUT\_LINE(staff\_cur.staff\_code||' '||staff\_cur.staff\_name);

END LOOP;

END;

BEGIN

OPEN staff\_cv FOR SELECT \* from staff\_masters;

Proc2(staff\_cv);

CLOSE staff\_cv;

END;

/

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Reff Cursor-Weak Type\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE PROCEDURE PROC1

IS

TYPE staffcurtype IS REF CURSOR ;

staff\_cv staffcurtype; --Declare cursor variable

PROCEDURE Proc2(p\_staff\_cv staffcurtype)

AS

staff\_cur department\_masters%rowtype;

BEGIN

LOOP

EXIT WHEN staff\_cv%NOTFOUND;

FETCH staff\_cv INTO staff\_cur;

DBMS\_OUTPUT.PUT\_LINE(staff\_cur.dept\_code||' '||staff\_cur.dept\_name);

END LOOP;

END;

BEGIN

OPEN staff\_cv FOR SELECT \* from department\_masters;

Proc2(staff\_cv);

CLOSE staff\_cv;

END;

**/**

Sql>execute proc1

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Exception Handling\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_staffno staff\_masters.staff\_code%type;

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

BEGIN

SELECT staff\_name into v\_name FROM staff\_masters WHERE staff\_code=&v\_staffno;

DBMS\_OUTPUT.PUT\_LINE(v\_name);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NOT FOUND');

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pragma Exception\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

v\_bookno number:=10000008;

child\_rec\_found EXCEPTION;

PRAGMA EXCEPTION\_INIT(child\_rec\_found,-2292);

BEGIN

DELETE FROM book\_masters WHERE book\_code=v\_bookno;

EXCEPTION

WHEN child\_rec\_found THEN

INSERT INTO error\_log VALUES ('BOOK entries exist for book'|| v\_bookno);

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Raise Exception\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

dup\_deptno EXCEPTION;

v\_counter binary\_integer;

v\_department number(2):=50;

BEGIN

SELECT count(\*) INTO v\_counter FROM department\_masters WHERE dept\_code=50;

DBMS\_OUTPUT.PUT\_LINE(v\_counter);

IF v\_counter >0 THEN

RAISE dup\_deptno;

END IF;

INSERT INTO department\_masters VALUES (v\_department,'new name');

EXCEPTION

WHEN dup\_deptno THEN

INSERT into error\_log VALUES ('DEPT:''||v\_departement|| already exist');

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Other Exception\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

v\_dummy varchar2(1) ;

v\_designation number(5):=111;

Err\_Num number(6);

Err\_Msg varchar2(10);

BEGIN

SELECT 'x' INTO v\_dummy FROM designation\_masters WHERE design\_code=v\_designation;

INSERT INTO error\_LOG VALUES('Designation:'||'already exist');

EXCEPTION

WHEN no\_data\_found THEN

INSERT into designation\_masters VALUES (v\_designation,'ITDEPT');

WHEN OTHERS THEN

Err\_Num:=SQLCODE;

Err\_Msg:=SUBSTR(SQLERRM,1,100);

INSERT INTO errors VALUES(err\_num,err\_msg);

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Pragma –Exception\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

CURSOR dept\_staff\_details\_cur

IS SELECT staff\_code,staff\_name

FROM staff\_masters

WHERE dept\_code=&dcode;

TYPE staff\_rec IS RECORD(

v\_staff\_code staff\_masters.staff\_code%TYPE,

v\_staff\_name staff\_masters.staff\_name%TYPE

);

v\_staff\_rec staff\_rec;

no\_staff\_exc EXCEPTION;

PRAGMA EXCEPTION\_INIT(no\_staff\_exc,-20888)

v\_code NUMBER;

v\_msg VARCHAR2(100);

BEGIN

IF(NOT dept\_staff\_details\_cur%ISOPEN) THEN

OPEN dept\_staff\_details\_cur;

END IF;

FETCH dept\_staff\_details\_cur INTO v\_staff\_rec;

IF(dept\_staff\_details\_cur%ROWCOUNT>0) THEN

LOOP

DBMS\_OUTPUT.PUT\_LINE(' Code :'|| v\_staff\_rec.v\_staff\_code

||

'Name :' ||v\_staff\_rec.v\_staff\_name);

FETCH dept\_staff\_details\_cur%NOTFOUND;

EXIT WHEN dept\_staff\_details\_cur%NOTFOUND;

END LOOP;

ELSE

--RAISE no\_staff\_exc;

RAISE\_APPLICATION\_ERROR(-20888,'No dept data exists in staff\_ma

ster');

END IF;

EXCEPTION

WHEN no\_staff\_exc THEN

DBMS\_OUTPUT.PUT\_LINE('Department Do Not Exist');

v\_code:=SQLCODE;

v\_msg:=SQLERRM;

INSERT INTO errorLog VALUES(emp\_id\_seq.NEXTVAL,v\_code,v\_msg,SYSDATE);

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Procedure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE PROCEDURE split\_name

( phrase IN VARCHAR2,p\_first OUT VARCHAR2,p\_last OUT VARCHAR2)

IS

BEGIN

p\_first := SUBSTR(phrase,1,INSTR(phrase,' ')-1);

p\_last := SUBSTR(phrase,INSTR(phrase,' ')+1);

IF p\_first = 'John' THEN

DBMS\_OUTPUT.PUT\_LINE('That is a common first name');

END IF;

END;

/

CREATE OR REPLACE PROCEDURE

Get\_Details(s\_code IN NUMBER,s\_name OUT varchar2, s\_sal OUT number ) IS

BEGIN

SELECT staff\_name,staff\_sal INTO s\_name,s\_sal FROM staff\_masters WHERE staff\_code=s\_code;

EXCEPTION

WHEN no\_data\_found THEN

INSERT INTO error\_log VALUES('NO Employee With ID'|| s\_code);

s\_name:=null;

s\_sal:=null;

END get\_details;

variable salary number

variable name varchar2

execute get\_details(100003,: name, :salary);

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Excecution of procedure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

s\_no number(10):=&sno;

sname varchar2(10);

sal number(10,2);

BEGIN

GET\_Details(s\_no,sname,sal);

dbms\_output.put\_line('Name:'||sname||'salary'||sal);

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Default Value In procedure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE PROCEDURE c\_dept(new\_dept IN NUMBER,new\_dname IN VARCHAR2 DEFAULT 'TEMP') IS

BEGIN

INSERT INTO department\_masters VALUES(new\_dept,new\_dname);

END;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE FUNCTION get\_AVG\_Sal (p\_deptno IN NUMBER) RETURN number AS

v\_sal number;

BEGIN

SELECT TRUNC(AVG(staff\_Sal)) INTO v\_sal FROM staff\_masters WHERE dept\_code=p\_deptno;

IF v\_sal IS NULL THEN

v\_sal:=-1;

END IF;

return v\_sal;

Exception

WHEN others THEN

return -2 ;--signifies any other error;

END get\_AVG\_Sal;

SQL> variable flag number;

SQL> execute :flag:=get\_avg\_sal(30);

SQL> print flag;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PACKAGES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

SQL> CREATE OR REPLACE PACKAGE Pack1 AS

Procedure Proc1;

Function Fun1 return varchar2;

End Pack1;

Package created.

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

**.**

SQL> CREATE OR REPLACE PACKAGE BODY pack1 AS

PROCEDURE Proc1 IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hi A Message From Procedure');

END Proc1;

FUNCTION Fun1 return VARCHAR2 IS

BEGIN

return('Hello From Function 1');

END Fun1;

END pack1;

Package body created.

SQL> set serverout on;

SQL> execute pack1.proc1;

Hi A Message From Procedure

**OR**

select pack1.fun1 from dual

CREATE OR REPLACE PACKAGE staff\_data as

TYPE staffcurtyp IS ref cursor return staff\_masters%rowtype;

Procedure Open\_Staff\_Cur(staff\_cur IN OUT staffcurtyp);

END staff\_data;

/

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Autonomous Transactions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

create table log1(sn number(10),sm varchar2(10));

desc log1;

TRUNCATE TABLE log1;

select \* from log1;

show errors;

CREATE Or Replace PROCEDURE Log\_Usage ( staff\_no IN number,

msg\_in IN varchar2)

IS

PRAGMA AUTONOMOUS\_TRANSACTION;

BEGIN

INSERT into log1 VALUES (staff\_no, msg\_in);

commit;

END LOG\_USAGE;

/

CREATE OR Replace PROCEDURE Chg\_Emp

IS

BEGIN

Log\_Usage(7566,'Chg sal ');

UPDATE log1 SET sm = 'hhh' WHERE sn=7566;

END chg\_emp;

/

INSERT INTO log1 VALUES(1001,'Hello');

exec Chg\_Emp();

select \* FROM log1;

rollback;

select \* FROM log1;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PL/SQL Supplied packages\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

DECLARE

BEGIN

DBMS\_OUTPUT.PUT('I am');

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

DBMS\_OUTPUT.PUT('BOOK');

END;

SQL> /

I am Writting

PL/SQL procedure successfully completed.

[We don’t see BOOK in output because it is not flushed]

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Get\_Line\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE FUNCTION get\_next\_line RETURN VARCHAR2 IS

return\_value VARCHAR2(255);

get\_status INTEGER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(444);

DBMS\_OUTPUT.GET\_LINE(return\_value,get\_status);

IF get\_status=0 THEN

RETURN return\_value;

ELSE

RETURN 'NULL';

END IF; END;

Sql>variable temp varchar2(255);

Sql >execute :temp:=get\_nine;

Print temp;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*File package\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

create directory sample as 'C:\Oracle\_Sample'

Note: Please create a folder physically in the Oracle server

Set serverout on

declare

f utl\_file.file\_type;

a varchar2(30);

begin

f :=utl\_file.fopen('SAMPLE','sample.txt','R');

for sctr in 1..3

loop

utl\_file.get\_line(f,a);

dbms\_output.put\_line(a);

end loop;

utl\_file.fclose(f);

end;

172.28.40.3 – C:\Oracle\_Sample\Sample.tx

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Trigger Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Where,

* CREATE [OR REPLACE] TRIGGER trigger\_name: Creates or replaces an existing trigger with the *trigger\_name*.
* {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
* {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
* [OF col\_name]: This specifies the column name that would be updated.
* [ON table\_name]: This specifies the name of the table associated with the trigger.
* [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
* [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
* WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

SQL> CREATE OR REPLACE TRIGGER display\_salary\_changes

BEFORE DELETE OR INSERT OR UPDATE

ON staff\_masters

FOR EACH ROW

WHEN (NEW.staff\_code > 0)

DECLARE

sal\_diff number;

BEGIN

sal\_diff := :NEW.staff\_sal - :OLD.staff\_sal;

dbms\_output.put\_line('Old salary: ' || :OLD.staff\_sal);

dbms\_output.put\_line('New salary: ' || :NEW.staff\_sal);

dbms\_output.put\_line('Salary difference: ' || sal\_diff);

END;

Trigger created.

SQL> INSERT INTO staff\_masters(staff\_code,staff\_name,staff\_sal) 2 VALUES (100,'vvv',1000);

Old salary:

New salary: 1000

Salary difference:

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

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ref Cursor Complex Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE PACKAGE pkg\_igate AS

TYPE StaffRefCurType is REF CURSOR RETURN staff\_masters%ROWTYPE;

PROCEDURE getStaffDetailsForDept(i\_dept\_code IN NUMBER,o\_staff\_cur OUT StaffRefCurType );

FUNCTION getStaffDetails(i\_staff\_cur IN StaffRefCurType )return StaffRefCurType ;

END pkg\_igate;

Package created.

CREATE OR REPLACE PACKAGE BODY pkg\_igate AS

PROCEDURE getStaffDetailsForDept(i\_dept\_code IN NUMBER, o\_staff\_cur OUT StaffRefCurType )AS

v\_staff\_cursor StaffRefCurType ;

BEGIN

OPEN v\_staff\_cursor FOR SELECT \* FROM staff\_masters WHERE dept\_code=i\_dept\_code ;

o\_staff\_cur := getStaffDetails(v\_staff\_cursor );

END getStaffDetailsForDept;

FUNCTION getStaffDetails(i\_staff\_cur IN StaffRefCurType )return StaffRefCurType AS

v\_staff\_cur StaffRefCurType ;

v\_staff\_details staff\_masters%ROWTYPE;

BEGIN

v\_staff\_cur :=i\_staff\_cur;

--LOOP

--FETCH i\_staff\_cur INTO v\_staff\_details ;

-- EXIT WHEN i\_staff\_cur%NOTFOUND;

-- DBMS\_OUTPUT.PUT\_LINE('...'||v\_staff\_details .staff\_code);

-- END LOOP;

-- CLOSE i\_staff\_cur;

return(v\_staff\_cur) ;

END getStaffDetails;

END pkg\_igate;

set serveroutput on;

variable b\_sd REFCURSOR;

execute pkg\_igate.getStaffDetailsForDept(10,:b\_sd );

print b\_sd ;

\*\*\*\*\*\*\*\*Parameter cursor\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

declare

cursor staffCur(dno number) is select

sm.empno,sm.ename,sm.esal,sm.deptno

from employee sm,dept dm

where sm.deptno=dm.deptno

and sm.deptno=dno; -- formal parameter

staffRec staffCur%Rowtype;

begin

open staffCur(10); -- actual parameter

loop

fetch staffCur into staffRec;

exit when staffCur%notfound;

dbms\_output.put\_line(staffRec.empno|| ' '||staffRec.ename||' '|| staffRec.esal);

end loop;

close staffCur;

end;

**\*\*\*\*\*\*\*class pkg demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE PACKAGE IGATE\_PKG AS

PROCEDURE getStaffDetails(

i\_staff\_code IN staff\_masters.staff\_code%TYPE,

o\_staff\_name OUT staff\_masters.staff\_name%TYPE,

o\_staff\_sal OUT staff\_masters.staff\_sal%TYPE);

FUNCTION calcAvgSal(i\_dept\_no IN staff\_masters.dept\_code %TYPE)RETURN NUMBER;

END IGATE\_PKG;

++++++++++++++++++++++++++++++++++++++++++++++++++++++=

CREATE OR REPLACE PACKAGE BODY IGATE\_PKG As

PROCEDURE getStaffDetails

(i\_staff\_code IN staff\_masters.staff\_code%TYPE,

o\_staff\_name OUT staff\_masters.staff\_name%TYPE,

o\_staff\_sal OUT staff\_masters.staff\_sal%TYPE)

AS

negative\_staff\_code EXCEPTION;

v\_err\_code NUMBER(10);

v\_err\_msg VARCHAR2(100);

BEGIN

IF(i\_staff\_code<0) THEN

RAISE negative\_staff\_code;

END IF;

SELECT staff\_name,staff\_sal

INTO o\_staff\_name ,o\_staff\_sal

FROM staff\_masters

WHERE staff\_code=i\_staff\_code;

EXCEPTION

WHEN negative\_staff\_code THEN

DBMS\_OUTPUT.PUT\_LINE(' Please enter non'||

'negative staff code');

WHEN no\_data\_found THEN

v\_err\_code:=SQLCODE;

v\_err\_msg:=SUBSTR(SQLERRM,1,20);

DBMS\_OUTPUT.PUT\_LINE(i\_staff\_code ||

' Does Not Exist ');

WHEN others THEN

v\_err\_code:=SQLCODE;

v\_err\_msg:=SUBSTR(SQLERRM,1,20);

DBMS\_OUTPUT.PUT\_LINE(v\_err\_code || ' '|| v\_err\_msg);

END getStaffDetails;

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION calcAvgSal(

i\_dept\_no IN staff\_masters.dept\_code%TYPE)

RETURN NUMBER

AS

v\_avg\_sal NUMBER(20,2);

v\_err\_no NUMBER(20);

v\_err\_msg VARCHAR(100);

BEGIN

SELECT AVG(staff\_sal) INTO v\_avg\_sal

FROM staff\_masters

WHERE dept\_code =i\_dept\_no;

IF(v\_avg\_sal IS NULL) THEN

RETURN 0.0;

ELSE

RETURN v\_avg\_sal;

END IF;

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE(' No Dept Code exist');

WHEN others THEN

v\_err\_no:=SQLCODE;

v\_err\_msg:=SUBSTR(SQLERRM,1,20);

DBMS\_OUTPUT.PUT\_LINE(' Other Errors '||

v\_err\_no ||' '||

v\_err\_msg);

END calcAvgSal;

END IGATE\_PKG;

++++++++++++++++++++++++++++++++++++++++++++++++++++++

DECLARE

v\_sname staff\_masters.staff\_name%TYPE;

v\_ssal staff\_masters.staff\_sal%TYPE;

v\_avgsal NUMBER(20,2);

BEGIN

IGATE\_PKG.getStaffDetails(10001,v\_sname,v\_ssal);

DBMS\_OUTPUT.PUT\_LINE(v\_sname || ' '||

v\_ssal);

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

v\_avgsal:= IGATE\_PKG.calcAvgSal(20);

DBMS\_OUTPUT.PUT\_LINE('Avg Salry Is '||

v\_avgsal);

END;

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ref Cursor Extra Demo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

CREATE OR REPLACE PACKAGE pkg\_igate AS

TYPE StaffRefCurType is REF CURSOR RETURN staff\_masters%ROWTYPE;

PROCEDURE getStaffDetailsForDept(i\_dept\_code IN NUMBER,o\_staff\_cur OUT StaffRefCurType );

FUNCTION getStaffDetails(i\_staff\_cur IN StaffRefCurType )return StaffRefCurType ;

END pkg\_igate;

Package created.

CREATE OR REPLACE PACKAGE BODY pkg\_igate AS

PROCEDURE getStaffDetailsForDept(i\_dept\_code IN NUMBER, o\_staff\_cur OUT StaffRefCurType )

AS

v\_staff\_cursor StaffRefCurType ;

BEGIN

OPEN v\_staff\_cursor FOR SELECT \* FROM staff\_masters WHERE dept\_code=i\_dept\_code ;

o\_staff\_cur := getStaffDetails(v\_staff\_cursor );

END getStaffDetailsForDept;

FUNCTION getStaffDetails(i\_staff\_cur IN StaffRefCurType )return StaffRefCurType AS

v\_staff\_cur StaffRefCurType ;

v\_staff\_details staff\_masters%ROWTYPE;

BEGIN

v\_staff\_cur :=i\_staff\_cur;

--LOOP

--FETCH i\_staff\_cur INTO v\_staff\_details ;

-- EXIT WHEN i\_staff\_cur%NOTFOUND;

-- DBMS\_OUTPUT.PUT\_LINE('...'||v\_staff\_details .staff\_code);

-- END LOOP;

-- CLOSE i\_staff\_cur;

return(v\_staff\_cur) ;

END getStaffDetails;

END pkg\_igate;

set serveroutput on;

variable b\_sd REFCURSOR;

execute pkg\_igate.getStaffDetailsForDept(10,:b\_sd );

print b\_sd ;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RefCursor-Pkg-Extra Demo-2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**####PACKAGE SPECIFICATION:**

CREATE OR REPLACE PACKAGE CG\_PKG

AS

TYPE dept\_ref\_cursor IS REF CURSOR;

-----type of the cursor is ref cursor variable

PROCEDURE getAllDept(deptCur IN OUT dept\_ref\_cursor);

END CG\_PKG;

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

CREATE OR REPLACE PACKAGE BODY CG\_PKG

AS

PROCEDURE getAllDept(deptCur IN OUT dept\_ref\_cursor)

AS

BEGIN

OPEN deptCur FOR SELECT \* FROM

department\_masters;

END getAllDept;

END CG\_PKG;

VARIABLE deptDetails REFCURSOR;

CALL CG\_PKG.getAllDept(:deptDetails);

PRINT deptDetails;