**Activity 4.1**

**Description :** **Views**

* **Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and corresponding values of calculated area in an empty table named Areas, consisting of two columns Radius and Area.**

CREATE TABLE AREAS(RADIUS INT,AREA NUMBER(10,2));

TABLE CREATED.

DECLARE

R NUMBER;

A NUMBER(10,2);

BEGIN

R:=&RADIUS;

LOOP

A:=R\*R\*3.14;

INSERT INTO AREAS VALUES(R,A);

R:=R+1;

EXIT WHEN R=8;

END LOOP;

END;

/

@ D:\area.sql;

area of circle with radius 3=28.26

area of circle with radius 4=50.24

area of circle with radius 5=78.5

area of circle with radius 6=113.04

area of circle with radius 7=153.86

PL/SQL procedure successfully completed.

SQL> select \* from areas;

RADIUS AREA

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

3 28.26

4 50.24

5 78.5

6 113.04

7 153.86

* **Write a PL/SQL block of code for inverting a number accepted from the console.**

DECLARE

NUM1 INT;

NUM2 INT;

BEGIN

NUM1:=&NUM1;

NUM2:=0;

LOOP

NUM2:=NUM2\*10+MOD(NUM1,10);

NUM1:=NUM1/10;

EXIT WHEN NUM1=0;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('REVERSE IS  '||NUM2);

END;

/

@ D:\reverse.sql;

Enter value for num1: 123

old   5: num1:=&num1;

new   5: num1:=123;

reverse is  321

* **Write a PL/SQL code block that will accept an account number from the user and debit an amount of Rs.2000 from the account if the account has a minimum balance of 500 after the amount is debited. The process is fired on the Accounts table.**

DECLARE

BALANCE NUMBER;

ACC\_NO VARCHAR2(10);

BEGIN

ACC\_NO:=&ACC\_NO;

SELECT BALANCE\_AMOUNT INTO BALANCE FROM ACCOUNT WHERE ACCOUNT\_NO=ACC\_NO;

IF BALANCE-2000<500 THEN

DBMS\_OUTPUT.PUT\_LINE('TRANSCATION IS NOT POSSIBLE');

ELSE

UPDATE ACCOUNT SET BALANCE\_AMOUNT=BALANCE\_AMOUNT-2000 WHERE ACCOUNT\_NO=ACC\_NO;

DBMS\_OUTPUT.PUT\_LINE('TRANSACTION COMPLETED');

END IF;

END;

/

* **Write a PL/SQL block of code that updates the salaries of Maria Jacob and Albert by Rs. 2000/- and Rs.2500/- respectively. Then check to see that the total salary does not exceed 75000. If the total salary is greater than 75000, then undo the updates made to salaries of both. (Use savepoint, rollback and commit).**

DECLARE

TOTAL NUMBER;

BEGIN

UPDATE BONUS SET SALARY  = SALARY + 2000  WHERE ENAME = 'MARIYA JACOB';

UPDATE BONUS SET SALARY  = SALARY + 2500  WHERE ENAME = 'ALBERT';

SELECT SUM(SALARY) INTO TOTAL FROM BONUS;

IF(TOTAL >=75000) THEN

DBMS\_OUTPUT.PUT\_LINE('TOTAL EXCEEDED. SO ROLLING BACK');

  ROLLBACK TO S2;

ELSE

  DBMS\_OUTPUT.PUT\_LINE('SALARIES UPDATED');

 SAVEPOINT S2;

END IF;

END;

/

**Activity 4.2**

**Description :** **Illustration of Cursors**

**Illustration of Implicit cursor**.

* **Write a PL/SQL block to accept an employee number and update the salary of that employee to raise the salary by 0.15. Display appropriate message based on the existence of the record in the employee table.**

DECLARE

BEGIN

UPDATE EMPLOYEE SET SALARY=&SALARY WHERE EMPNO=&EMPNO;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT||'RECORDS ARE UPDATED IN EMPLOYEETABLE');

END IF;

END;

/

* **The HRD manager decides to raise the salary of employees working as ‘analyst’ by 0.15. Write a cursor to update the salary of the employees. Display the no. of employee records that has been modified.**

DECLARE

BEGIN

UPDATE EMPLOYEEE1 SET SALARY=SALARY+SALARY\*0.15 WHERE JOB='ANALYST';

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT||'RECORDS ARE UPDATED');

END IF;

END;

/

**Illustration of explicit cursor.**

* **Write an explicit cursor to display the name,department, salary of the first 5 employees getting the highest salary.**

DECLARE

CURSOR EMPCURSOR IS SELECT EMPNAME,DEPTNAME,SALARY FROM EMPLOYEEE1 E,DEPARTMENT2 D WHERE E.DEPTNO=D.DEPTNO ORDER BY SALARY DESC;

NAME EMPLOYEE.EMPNAME%TYPE;

DEPARTMENT DEPARTMENT.DEPTNAME%TYPE;

EMP\_SALARY EMPLOYEE.SALARY%TYPE;

BEGIN

OPEN EMPCURSOR;

WHILE EMPCURSOR%ROWCOUNT !=6

LOOP

FETCH EMPCURSOR INTO NAME,DEPARTMENT,EMP\_SALARY;

DBMS\_OUTPUT.PUT\_LINE('EMPNAME :'||NAME);

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

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

END LOOP;

CLOSE EMPCURSOR;

END;

/

empname :ARNOLD LEONARD AMON

department :COMPUTER SERVICE DIVISION

salary :152750

empname :DONA ANICE SIBY

department :COMPUTER SERVICE DIVISION

salary :46500

empname :PHILIP VINCENT

department :PLANNING

salary :41250

empname :ALFRIN LUIZ

department :SUPPORT SERVICES

salary :40175

empname :SHILVY K K

department :INFORMATION CENTER

salary :38250

empname :JEFFIN DOMINIC

department :INFORMATION CENTER

salary :37585.45

* **The HRD manager decides to raise the salary of employees working as ‘analyst’ by 0.15. Whenever any such raise is given to the employees, a record for the same is maintained in the emp\_raise table. It includes the employee number, the date when the raise was given and actual raise. Write a PL/SQL block to update the salary of the employees and insert a record in the emp\_raise table. Emp\_raise(empcode, raisedate,raise\_amt)**

CREATE TABLE EMP\_RAISE(EMPNO CHAR(10),RAISEDATE DATE,RAMOUNT NUMBER(10,2));

TABLE CREATED.

DECLARE

CURSOR C1 IS SELECT EMPNO,SALARY FROM EMPLOYEEE1 WHERE JOB='ANALYST';

ENO EMPLOYEEE1.EMPNO%TYPE;

ESAL EMPLOYEEE1.SALARY%TYPE;

RAMT NUMBER;

BEGIN

OPEN C1;

FETCH C1 INTO ENO,ESAL;

WHILE C1%FOUND

LOOP

     RAMT:= ESAL \* 0.15;

     UPDATE EMPLOYEEE1 SET SALARY = SALARY + RAMT WHERE EMPNO = ENO;

     INSERT INTO EMP\_RAISE VALUES(ENO,SYSDATE,RAMT);

     FETCH C1 INTO ENO,ESAL;

END LOOP;

CLOSE C1;

END;

/

@ D:\SQL\EXPLICITE6

PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

SELECT \*FROM EMP\_RAISE;

EMPNO  RAISEDATE RAMOUNT

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

E0130  11-OCT-23 4721.33

E0140  11-OCT-23 5637.82

E0130  11-OCT-23 5429.52

E0140  11-OCT-23 6483.49

E0130  11-OCT-23 6243.95

E0140  11-OCT-23 7456.01

6 rows selected.

**Activity 4.3**

**Description :** **Illustration of Procedures**

* **Write a PL/SQL block which makes use of a stored procedure Proj\_emp ( emp\_name varchar2(50) ) which finds all the details of the projects involved by the given employee.**

DECLARE

EMPNAME CHAR(30);

BEGIN

EMPNAME:=&EMPNAME;

PROJ\_EMP(EMPNAME);

END;

/

CREATE OR REPLACE PROCEDURE PROJ\_EMP(EMPNAME IN VARCHAR2) AS

CURSOR EMP IS SELECT PROJNAME FROM EMPLOYEEE1 E,PROJECT P,EMP\_PROJJ EP WHERE E.EMPNO=EP.EMPNO AND P.PROJNO=EP.PROJNO AND EMPNAME=EMPNAME;

PROJECT\_NAME PROJECT.PROJNAME%TYPE;

BEGIN

OPEN EMP;

LOOP

FETCH EMP INTO PROJECT\_NAME;

EXIT WHEN EMP%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('PROJECT NAME '||PROJECT\_NAME);

END LOOP;

CLOSE EMP;

END;

/

* **Write a procedure to check whether a string is a palindrome . Call the procedure to list all the palindrome names in the employee table.**

DECLARE

CURSOR FIND IS SELECT EMPNAME FROM EMPLOYEE;

ENAME VARCHAR(35);

BEGIN

OPEN FIND;

LOOP

FETCH FIND INTO ENAME;

EXIT WHEN FIND%NOTFOUND;

PAL(ENAME);

END LOOP;

CLOSE FIND;

END;

/

CREATE OR REPLACE PROCEDURE PAL(NAME IN VARCHAR2) as

REV VARCHAR(30);

BEGIN

SELECT REVERSE(NAME) INTO REV FROM DUAL;

IF NAME = REV

THEN

DBMS\_OUTPUT.PUT\_LINE(NAME);

END IF;

END;

/

* **Write a PL/SQL block which retrieve all the employee into a cursor and display the details of all assigned  projects for each employee using a stored procedure Proj\_emp ( emp\_name varchar2(50).**

CREATE OR REPLACE PROCEDURE PROJ\_EMP(ENAME IN VARCHAR2) AS

CURSOR EMPRO IS SELECT EP.PROJNO,PROJNAME FROM EMPLOYEE E INNER JOIN EMP\_PROJ EP ON EP.EMPNO=E.EMPNO INNER JOIN PROJECT P ON EP.PROJNO=P.PROJNO WHERE E.EMPNAME=ENAME;

PNA VARCHAR2(30);

PNO CHAR(10);

BEGIN

OPEN EMPRO;

LOOP

FETCH EMPRO INTO PNO, PNA;

EXIT WHEN EMPRO%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(PNO||PNA);

END LOOP;

CLOSE EMPRO;

END;

/

--execute Proj\_emp(&ename);

**Activity 4.4**

**Description :** **Illustration of functions**

* **Write a function to find the reverse of EmpNo in Employee table and display the EmpNo and Reversed(Emp No) of the first 5 employees using an SQL Query.**

DECLARE

CURSOR C1 IS SELECT EMPNO FROM EMPLOYEE WHERE ROWNUM <=5;

EMP C1%ROWTYPE;

REV1 CHAR(10);

BEGIN

OPEN C1;

FETCH C1 INTO EMP;

WHILE C1%FOUND

LOOP

REV1:=REV(EMP.EMPNO);

DBMS\_OUTPUT.PUT\_LINE(EMP.EMPNO||' '||REV1);

FETCH C1 INTO EMP;

END LOOP;

CLOSE C1;

END;

/

CREATE OR REPLACE FUNCTION

REV(EMPNO IN CHAR)

RETURN CHAR

IS

REV1 CHAR(10);

BEGIN

SELECT REVERSE(EMPNO) INTO REV1 FROM DUAL;

RETURN REV1;

END;

/

* **Write a function that would check for the existence of an employee in the employee table given an EmpNo. If existing employee, check whether he is the manager of any department and display messages accordingly.**

CREATE OR REPLACE FUNCTION CHK\_MANAGER(EMPNO IN VARCHAR2) RETURN VARCHAR2

AS

EMPNAME EMPLOYEE.EMP\_NAME%TYPE;

DEPTNAME DEPARTMENT.DEPT\_NAME%TYPE;

A NUMBER;

CURSOR EMP\_CUR IS SELECT EMP\_NAME,DEPT\_NAME FROM EMPLOYEE E INNER JOIN DEPARTMENT D ON E.DEPT\_NO=D.DEPT\_NO WHERE EMP\_NO=EMPNO AND JOB LIKE 'MANAGER%';

BEGIN

OPEN EMP\_CUR;

DBMS\_OUTPUT.PUT\_LINE(EMPNO);

SELECT LENGTH(EMPNO) INTO A FROM DUAL;

DBMS\_OUTPUT.PUT\_LINE(A);

FETCH EMP\_CUR INTO EMPNAME,DEPTNAME;

IF EMP\_CUR%NOTFOUND

THEN

RETURN 'EMPLOYEE NOT FOUND';

ELSE

DBMS\_OUTPUT.PUT\_LINE('EMPLOYEE FOUND');

FETCH EMP\_CUR INTO EMPNAME,DEPTNAME;

RETURN EMPNAME||' MANAGER OF '||DEPTNAME;

END IF;

CLOSE EMP\_CUR;

END;

/

DECLARE

EMPNO VARCHAR2(30);

BEGIN

EMPNO:=&EMPNO;

DBMS\_OUTPUT.PUT\_LINE(CHK\_MANAGER(EMPNO));

END;

/

**Activity 4.5**

**Description :** **Illustration of Triggers**

* **Consider the table Employee. Write PL/SQL statements to create a trigger when fired checks the operation performed on a table and based on the operation, a variable is assigned the value ‘update’ or ‘delete’. Previous values of the modified record of the table Employee are stored into the appropriate variables declared and inserted to the audit table AuditEmployee.**

CREATE OR REPLACE TRIGGER LABTRIG1 BEFORE UPDATE OR DELETE ON EMPLOYEE

FOR EACH ROW

OPERATION VARCHAR2(10);

BEGIN

IF UPDATING

THEN

OPERATION:='UPDATION';

INSERT INTO AUDITEMPLOYEE VALUES(:OLD.EMP\_NO,:OLD.EMP\_NAME,OPERATION);

ELSIF DELETING

THEN

OPERATION:='DELETION';

INSERT INTO AUDITEMPLOYEE VALUES(:OLD.EMP\_NO,:OLD.EMP\_NAME,OPERATION);

ELSIF INSERTING

THEN

OPERATION:='INSERTION';

INSERT INTO AUDITEMPLOYEE VALUES(:OLD.EMP\_NO,:OLD.EMP\_NAME,OPERATION);

END IF;

END;

/

* **Write PL/SQL statements to create a trigger which generates an error messages if the salary is below or beyond the valid range 0-5000 on the employee table. The triggering events are update and insert.**

CREATE OR REPLACE TRIGGER LABTRIG2 BEFORE UPDATE OR INSERT ON EMPLOYEE

FOR EACH ROW

BEGIN

IF UPDATING OR INSERTING

THEN

IF (:NEW.SALARY<0) OR (:NEW.SALARY > 5000)

THEN

RAISE\_APPLICATION\_ERROR(-20500,'INVALID SALARY');

END IF;

END IF;

END;

/

* **Write PL/SQL statements to create a trigger that limits the DML actions to the Employee table to weekdays from 8.30am to 6.30pm. If a user tries to insert/update/delete a row in the Employee table, a warning message will be prompted.**

CREATE OR REPLACE TRIGGER TRG1 BEFORE DELETE OR INSERT OR UPDATE ON emp\_tab

BEGIN

IF

(TO\_CHAR(SYSDATE, 'dy') IN ('sat', 'sun'))

 OR (TO\_CHAR(SYSDATE,'hh24:mi') NOT BETWEEN '08:30' AND '18:30')

THEN       RAISE\_APPLICATION\_ERROR(-20500, 'table is secured');

END IF;

END;

/