**PL-SQL\_Lab-3**

**(Cursors in PL/SQL)**

**Consider the table Student (Rollno, name, age, mark1, mark2, mark3, total). (For**

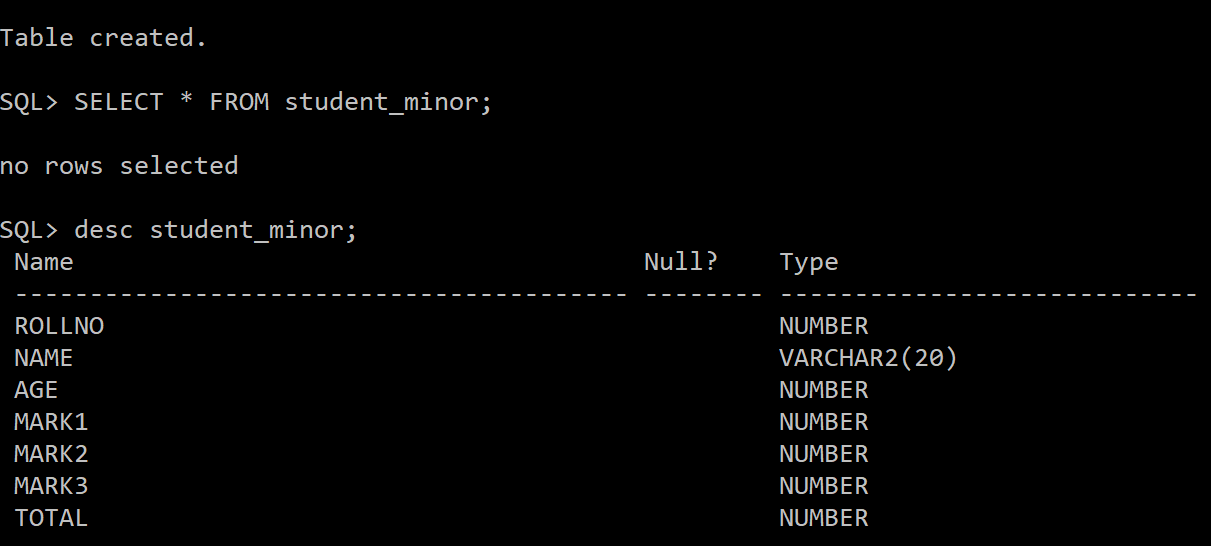
**Q1 to Q6)**

**1. Write a pl/sql code using cursor, which will delete all those records from the**

**Student table where age < 25 and insert those records into another table**

**called Student-minor.**

CREATE TABLE student\_minor AS SELECT \* FROM student WHERE 1=2;



DECLARE

CURSOR C1 IS SELECT \* FROM student WHERE AGE<25;

temp C1%ROWTYPE;

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO temp;

EXIT WHEN C1%NOTFOUND;

INSERT

INTO student\_minor VALUES(temp.rollno, temp.name,temp.age, temp.mark1, temp.mark2, temp.mark3, temp.total);

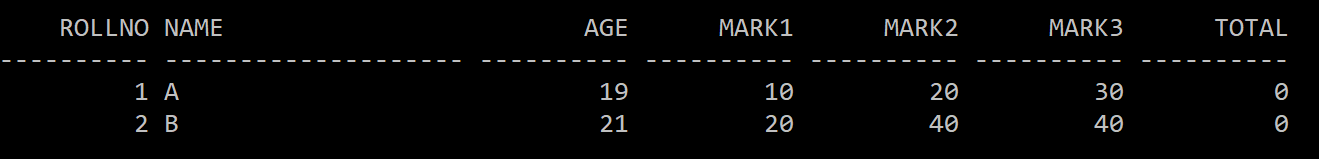
DELETE FROM student WHERE rollno=temp.rollno;

END LOOP;

CLOSE C1;

END;

/



A black screen with white text

Description automatically generated

**2. Write a pl/sql code using cursor, which will find the total marks of each**

**student and update the total column (assume that initially, the total is zero**

**for all the students).**

DECLARE

CURSOR C1 IS SELECT \* FROM student;

temp C1%ROWTYPE;

t student.total%TYPE;

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO temp;

EXIT WHEN C1%NOTFOUND;

SELECT temp.mark1+temp.mark2+temp.mark3 INTO t FROM STUDENT WHERE temp.rollno=student.rollno;

UPDATE student SET student.total=t WHERE temp.rollno=student.rollno;

END LOOP;

CLOSE C1;

END;

/

**A screenshot of a computer

Description automatically generated**

**3. Write a pl/sql code using cursor to find out how many students are there**

**whose total marks are greater than 90, and then display their details.**

DECLARE

c number:=0;

CURSOR C1 IS SELECT \* FROM STUDENT WHERE TOTAL>90;

BEGIN

FOR r IN C1 LOOP

EXIT WHEN C1%NOTFOUND;

SELECT \* INTO r FROM STUDENT WHERE rollno=r.rollno;

DBMS\_OUTPUT.PUT\_LINE(r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

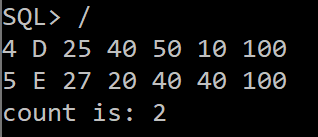
c:=c+1;

END LOOP;

dbms\_output.put\_line('count is: '||c);

END;

/



**4. Write a pl/sql code using cursor to find the highest and lowest marks and**

**display the corresponding student’s details.**

DECLARE

max\_marks NUMBER;

min\_marks NUMBER;

CURSOR C1 IS SELECT \* FROM STUDENT;

BEGIN

SELECT max(total) INTO max\_marks FROM student;

SELECT min(total) INTO min\_marks FROM student;

FOR R IN C1 LOOP

EXIT WHEN C1%NOTFOUND;

IF R.total=max\_marks THEN

DBMS\_OUTPUT.PUT\_LINE('MAX MARKS: '||r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

ELSIF R.total=min\_marks THEN

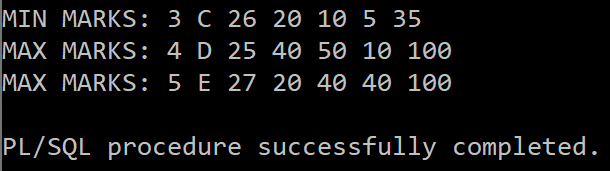
DBMS\_OUTPUT.PUT\_LINE('MIN MARKS: '||r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

END IF;

END LOOP;

END;

/



**5. Write a pl/sql code using cursor to find the average mark of all the students**

**and display it on the screen.**

DECLARE

a NUMBER(4,2);

CURSOR C1 IS SELECT \* FROM student;

BEGIN

FOR R IN C1 LOOP

EXIT WHEN C1%NOTFOUND;

SELECT (R.mark1+R.mark2+R.mark3)/3 INTO a FROM student WHERE rollno=R.rollno;

DBMS\_OUTPUT.PUT\_LINE('ROLLNO: '||R.rollno||' AVERAGE: '||a);

END LOOP;

END;

/

A screen shot of a black background

Description automatically generated

**6. Write a stored procedure using a parameterized cursor, which will display**

**the student details whose rollno is passing as a parameter to the cursor from**

**the stored procedure.**

CREATE OR REPLACE PROCEDURE f(n in number) AS

CURSOR C1 IS SELECT \* FROM student WHERE rollno=n;

BEGIN

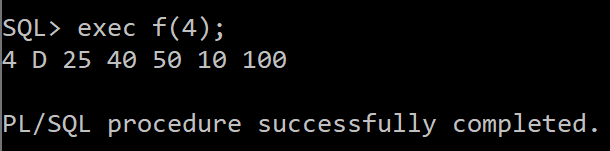
FOR R IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

END LOOP;

END**;**

**/**

****

**Consider the table EMP (empno, ename, job, sal, deptno) (For Q7 to Q10)**

**7. Write a PL/SQL code to demonstrate %TYPE and %ROWTYPE to display**

**details of employees in EMP table.**

**A black screen with white text

Description automatically generated**

DECLARE

CURSOR C1 IS SELECT \* FROM EMP WHERE EMPNO=1;

ENO EMP.EMPNO%TYPE;

R EMP%ROWTYPE;

BEGIN

FOR R IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(R.EMPNO||' '||R.ENAME||' '||R.JOB||' '||R.SAL||' '||R.DEPTNO);

SELECT EMPNO INTO ENO FROM EMP WHERE EMPNO=R.EMPNO;

DBMS\_OUTPUT.PUT\_LINE(ENO);

END LOOP;

END;

/

A black background with white text

Description automatically generated

**8. Write a stored function to display the empno, ename, and job of employees**

**of a department for EMP table using a parameterized cursor where deptno**

**will be sent as a parameter to the cursor from the stored function.**

CREATE OR REPLACE FUNCTION FUN(N IN NUMBER) RETURN NUMBER IS ANS NUMBER:=0;

CURSOR C1 IS SELECT \* FROM EMP WHERE EMPNO=N;

BEGIN

FOR R IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(R.EMPNO||' '||R.ENAME||' '||R.JOB);

END LOOP;

ANS:=1;

RETURN ANS;

END;

/

DECLARE

ANS NUMBER;

BEGIN

ANS:=FUN(2);

END;

/

A black background with white text

Description automatically generated

**9. Write a local function to display the employee number and name of the top**

**‘n’ highest-paid Employees using parameterized cursor. The value of ‘n’ is**

**passed to the cursor as a parameter from the local function.**

DECLARE

ANS NUMBER;

FUNCTION FUN(N IN NUMBER) RETURN NUMBER IS ANS NUMBER:=0;

CURSOR C1 IS SELECT \* FROM EMP;

ENO NUMBER;

BEGIN

FOR R IN C1 LOOP

SELECT EMPNO INTO ENO FROM EMP E1 WHERE N-1 = (SELECT COUNT(\*) FROM EMP E2 WHERE E2.SAL>E1.SAL);

IF R.EMPNO=ENO THEN

DBMS\_OUTPUT.PUT\_LINE(R.EMPNO||' '||R.ENAME);

END IF;

END LOOP;

ANS:=1;

RETURN ANS;

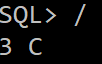
END;

BEGIN

ANS:=FUN(2);

END;

/



**10. Write a local procedure to calculate the total salary of the first ‘n’ records of**

**EMP table using parameterized cursor. The value of ‘n’ is passed to cursor**

**as a parameter from the local procedure.**

DECLARE

N NUMBER:=3;

S NUMBER:=0;

PROCEDURE FINDSUM(N IN NUMBER) IS

CURSOR C1 IS SELECT \* FROM EMP;

C NUMBER:=0;

BEGIN

FOR R IN C1 LOOP

EXIT WHEN C>=N OR C1%NOTFOUND;

S:=S+R.SAL;

C:=C+1;

END LOOP;

END;

BEGIN

FINDSUM(N);

DBMS\_OUTPUT.PUT\_LINE(S);

END;

/

A black background with white text

Description automatically generated

**(Exception Handling in PL/SQL)**

**11. Write a PL/SQL program to demonstrate the following exceptions:**

**When Too Many Rows**

**When No Data Found**

**When Others**

--too many rows exception

DECLARE

A NUMBER;

BEGIN

SELECT EMPNO INTO A FROM EMP;

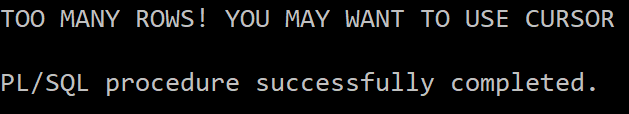
EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS! YOU MAY WANT TO USE CURSOR');

END;

/



--NO DATA FOUND ERROR

DECLARE

A NUMBER;

BEGIN

SELECT EMPNO INTO A FROM EMP WHERE EMPNO=5;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

END;

/

A black background with white text

Description automatically generated

--OTHERS

--HERE NO SUCH TABLE EXISTS

DECLARE

A NUMBER;

BEGIN

SELECT EMPNO INTO A FROM EMP1 WHERE EMPNO=5;

EXCEPTION

WHEN OTHERS THEN

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

END;

/

A black background with white text

Description automatically generated

**12. Write a PL/SQL program to demonstrate the User Defined Exceptions.**

DECLARE

A NUMBER;

B NUMBER;

C NUMBER;

EX EXCEPTION;

BEGIN

A:=&A;

B:=&B;

IF MOD(B,A)<>0 THEN

RAISE EX;

END IF;

C:=B/A;

DBMS\_OUTPUT.PUT\_LINE(C);

EXCEPTION

WHEN EX THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR! ENTER MULTIPLE OF FIRST NUMBER AS SECOND NUMBER');

END;

/

**A black screen with white text

Description automatically generated**

**A black screen with white text

Description automatically generated**