

LAB CYCLE 2

1. Write a PL/SQL code to calculate total and percentage of marks of a student in four subjects.

```
declare
    rollno number;
    mark1 number;
    mark2 number;
    mark3 number;
    mark4 number;
    total number;
    percentage number(8,2);
begin
    rollno:=&rollno;
    mark1:=&mark1;
    mark2:=&mark2;
    mark3:=&mark3;
    mark4:=&mark4;
    total:=mark1+mark2+mark3+mark4;
    percentage:=total*0.25;
    dbms_output.put_line('Student Marklist');
    dbms_output.put_line('Total Mark = '|| total);
    dbms_output.put_line('Percentage = '|| percentage);
End;
```

OUTPUT

Student Marklist
Total Mark = 110
Percentage = 27.5

PL/SQL procedure successfully completed.

```
Student Marklist  
Total Mark = 110  
Percentage = 27.5
```

```
PL/SQL procedure successfully completed.
```

2. Write a PL/SQL code to calculate the total and the percentage of marks of the students in four subjects from the table, STUDENT with the following schema.
STUDENT (RNO , S1 , S2, S3, S4, total, percentage).

```
declare  
    t student.total%type;  
    p student.percentage%type;  
    cursor STU is select * from student;  
    rw stu%rowtype;  
begin  
    open STU;  
    loop  
        fetch STU into rw;  
        exit when STU%notfound;  
        t:=rw.s1+rw.s2+rw.s3+rw.s4;  
        p:=t*0.25;  
        update student set total=t,percentage=p where Rno=rw.Rno;  
    end loop;  
    close stu;  
End;
```

OUTPUT

```
PL/SQL procedure successfully completed.
```

	RNO	S1	S2	S3	S4	TOTAL	PERCENTAGE
1	1	10	20	30	40	100	25
2	2	11	22	33	44	110	27.5

3. Write a PL/SQL code to calculate the total salary amount of the first n records of the employee table.

```

declare
    n number;
    i number:=1;
    tot number:=0;
    cursor emp is select salary from employees;
    cemp emp%rowtype;
begin
    n:=5;
    open emp;
    while (i<=n)
    loop
        fetch emp into cemp;
        tot:=tot+cemp.salary;
        i:=i+1;
    end loop;
    dbms_output.put_line('Total salary of '||n||' is '||tot);
    close emp;
end;
```

OUTPUT

Total salary of 5 is 113275

PL/SQL procedure successfully completed.

```
Total salary of 5 is 113275
```

```
PL/SQL procedure successfully completed.
```

3. Write a PL/SQL code to calculate the total salary amount of the first n records of the employee table.

```
declare
    n number;
    i number:=1;
    tot number:=0;
    cursor emp is select salary from employees;
    cemp emp%rowtype;
begin
    n:=5;
    open emp;
    while (i<=n)
    loop
        fetch emp into cemp;
        tot:=tot+cemp.salary;
        i:=i+1;
    end loop;
    dbms_output.put_line('Total salary of '||n||' is '||tot);
    close emp;
end;
```

OUTPUT

Total salary of 5 is 113275

PL/SQL procedure successfully completed.

```
Total salary of 5 is 113275
```

```
PL/SQL procedure successfully completed.
```

4. Use Cursors and add a user-defined exception to raise an exception if the number of

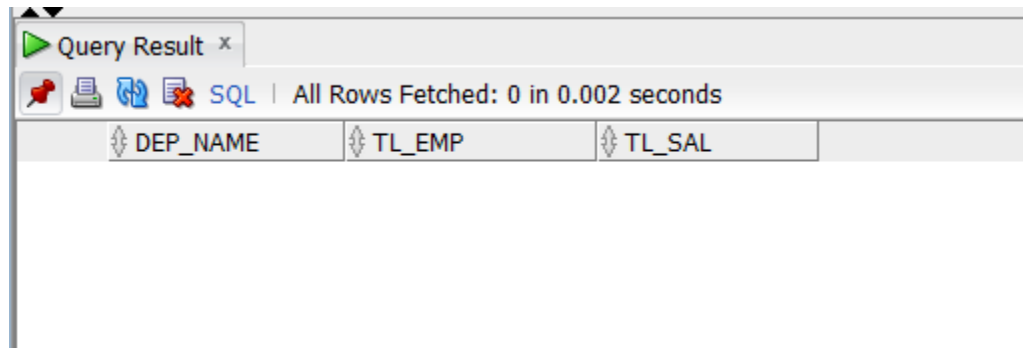
employees in a particular department is less than 2. If the number of employees is less than 2, then print a message 'Department status needs 2 or more employees'. If the number is greater than 2, then populate the Department_stat table (dname, tot_emps, tot_salary).

```
declare
    dep_id employees.department_id%type;
    cursor dep is select * from employees where department_id = dep_id;
    rw dep%rowtype;
    tl_emp number:=0;
    tl_sal number:=0;
    dep_name varchar(15);
    execp exception;
begin
    dep_id:=&dep_id;
    open dep;
    select department_name into dep_name from departments where department_id =
dep_id;
    loop
        fetch dep into rw;
        exit when dep%notfound;
        tl_emp:=tl_emp+1;
        tl_sal := tl_sal + rw.salary;
    end loop;
    if(tl_emp<2) then
        raise execp;
    else
        insert into department_stat values(dep_name , tl_emp , tl_sal);
    end if;
    close dep;
exception
    when execp then
        dbms_output.put_line('dep status needs 2 or more employees');
end;
```

OUTPUT

dep status needs 2 or more employees

PL/SQL procedure successfully completed.



5. Write a PL/SQL procedure to perform the concatenation of two strings. Strings need to be accepted through parameter passing.

```
create or replace procedure c_string(str1 in varchar, str2 in varchar) as
    str3 varchar(20);
begin
    str3:=CONCAT(str1,str2);
    dbms_output.put_line('concatenated string is:'||str3);
end;
```

```
declare
    str1 varchar(20);
    str2 varchar(20);
begin
    str1:='Anila';
    str2:='Mathew';
    c_string(str1,str2);
end;
```

```
create or replace procedure c_string(str1 in varchar, str2 in varchar) as
    str3 varchar(20);
begin
    str3:=CONCAT(str1,str2);
    dbms_output.put_line('concatenated string is:'||str3);
end;
```

```
/
accept str1 prompt 'enter the value of frist string:';
accept str2 prompt 'enter the value of second string:';
declare
    str1 varchar(20);
    str2 varchar(20);
begin
    str1:='&str1';
    str2:='&str2';
    c_string(str1,str2);
end;
/
```

OUTPUT

concatenated string is:amma appa

PL/SQL procedure successfully completed.

```
concatenated string is:amma appa

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL procedure to find the number of students ranging from 100-70%, 69-60%, 59-50% & below 49% from the STUDENT table.

```
declare
cursor cur_stud is select percentage as p from student;
c70 int;
c60 int;
c50 int;
c49 int;
```

```

rw cur_stud%rowtype;
begin
c70:=0;
c50:=0;
c60:=0;
c49:=0;
open cur_stud;
loop
    fetch cur_stud into rw;
    exit when cur_stud%notfound;
    if (rw.p >=70) and (rw.p<=100) then
        c70:=c70+1;
    else if (rw.p >=60) and (rw.p<=69) then
        c60:=c60+1;
    else if (rw.p >=50) and (rw.p<=59) then
        c50:=c50+1;
    else
        c49:=c49+1;
    end if;
    end if;
    end if;
end loop;
close cur_stud;
dbms_output.put_line('students with percentage 100-70 ' ||c70);
dbms_output.put_line('students with percentage 69-60 ' ||c60);
dbms_output.put_line('students with percentage 59-50 ' ||c50);
dbms_output.put_line('students with percentage below 49 ' ||c49);
end;

```

OUTPUT

```

students with percentage 100-70 0
students with percentage 69-60 0
students with percentage 59-50 0
students with percentage below 49 2

```

PL/SQL procedure successfully completed.


```
students with percentage 100-70 0
students with percentage 69-60 0
students with percentage 59-50 0
students with percentage below 49 2
```

PL/SQL procedure successfully completed.

7.Create a function that accepts a number and returns its reverse value. Also write the program to invoke this function.

```
declare
    a int;
    c int;
    n int;
    rev int:=0;
    r int;
    function reverse_it( x IN int) return int as z int;
    begin
        n := x;
        while (n > 0)
        loop
            r := mod(n , 10);
            rev := (rev * 10) + r;
            n := n / 10;
        end loop;
        z := rev;
        return z;
    end ;
BEGIN
    a := &a;
    c := reverse_it(a);
```

```
        dbms_output.put_line('the reverse of number is ' || c);  
END;
```

OUTPUT

The number is 23
the reverse of number is 32

PL/SQL procedure successfully completed.

```
The number is 23  
the reverse of number is 32  
  
PL/SQL procedure successfully completed.
```

9. Write a row trigger to add the details of new employees in Newemployee table, relieved

employees in DelEmployee table and updated employees in ModiEmployee table. Trigger need to be fired after the insertion/deletion/updation made with Employee table.

```
CREATE OR REPLACE TRIGGER mytrig2  
    AFTER DELETE OR INSERT OR UPDATE ON employee19  
    FOR EACH ROW BEGIN IF DELETING THEN  
        INSERT INTO delemmployee19(ename,city) VALUES (:old.ename, :old.city);  
    ELSE  
        INSERT INTO modiemployee19(ename,city) VALUES (:new.ename, :old.city);  
    END IF;  
END
```

```
select * from ModiEmployee;
```

Script Output x		Query Result x	
SQL All Rows Fetched: 15 in 0.003 seconds			
	ENAME	CITY	
1	unni	kochi	
2	unni	kochi	
3	ammu	tvpm	
4	ammu	tvpm	