DATABASE MANAGEMENT SYSTEM LAB

THIAGARAJAR COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

B.Sc., (Computer Science with Cognitive Systems)

(Self Finance)

NAME :

REGISTER NO :

COURSE CODE :

COURSE TITLE :

THIAGARAJAR COLLEGE(AUTONOMOUS)

(Affiliated to Madurai Kamaraj University)
Re-Accredited with 'A++' Grade by NAAC
Ranked 18th in NIRF 2023

B.Sc., Computer Science with Cognitive Systems

(Self Finance)

Certified that this is the bonafide record of practical work done by _____ ***Reg.No.*** _____ ***for the*** _____

laboratory during the Academic Year 2023 - 24.

Submitted for the External practical Examination held on _____.

Internal Examiner **External Examiner**

Cursor			
27.		implicit cursor	
28.		explicit cursor using basic loop	
29.		explicit cursor using for loop	
Trigger			
30.		Age checking	
31.		Case conversion	
Package			
32.		Arithmetic operations	

Signature of the faculty

EX.NO: 1 ARITHMETIC OPERATORS

```
SQL> select 10 + 10 as Addition from dual;

ADDITION
----------
        20

SQL> select 30 - 10 as subtraction from dual;

SUBTRACTION
-----------
         20

SQL> select 30 * 10 as multiplication from dual;

MULTIPLICATION
--------------
           300

SQL> select 30 / 10 as division from dual;

DIVISION
----------
         3
```

RESULT:

The arithmetic operators are executed successfully and thus the expected output is obtained.

EX.NO:2 COMPARISON OPERATORS

SQL> select * from employee;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
102	mano	12	20000
103	priya	12	15000
104	shiva	14	14000
105	sree	10	20000

SQL> select * from employee where salary < 16000;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
103	priya	12	15000
104	shiva	14	14000

SQL> select * from employee where salary > 16000;

ENO	ENAME	DEP	SALARY
102	mano	12	20000
105	sree	10	20000

SQL> select * from employee where salary = 15000;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
103	priya	12	15000

SQL> select * from employee where salary <> 15000;

ENO	ENAME	DEP	SALARY
102	mano	12	20000
104	shiva	14	14000
105	sree	10	20000

SQL> select * from employee where salary != 15000;

ENO	ENAME	DEP	SALARY
102	mano	12	20000
104	shiva	14	14000
105	sree	10	20000

SQL> select * from employee where salary <= 15000;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
103	priya	12	15000
104	shiva	14	14000

SQL> select * from employee where salary >= 15000;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
102	mano	12	20000
103	priya	12	15000
105	sree	10	20000

RESULT:

The comparison operators are executed successfully and thus the expected output is obtained.

EX.NO:3 LOGICAL OPERATORS

SQL> select * from employee;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
102	mano	12	20000
103	priya	12	15000
104	shiva	14	14000
105	sree	10	20000

SQL> select * from dept;

DEPTID	DEPTNAME
10	Production
11	marketing
12	sales
13	HR
14	Stock

SQL> select * from employee where salary > all(15000,10000,12000);

ENO	ENAME	DEP	SALARY
102	mano	12	20000
105	sree	10	20000

SQL> select * from employee where salary > all(select salary from employee where deptid=14);

```
    ENO ENAME        DEP    SALARY
---------- -------------- --- ----------
    102 mano         12      20000
    105 sree         10      20000
```

SQL> select * from employee where salary > any(15000,18000);

```
    ENO ENAME        DEP    SALARY
---------- -------------- --- ----------
    102 mano         12      20000
    105 sree         10      20000
```

SQL> select * from employee where salary > any(select salary from employee where deptid=12);

```
    ENO ENAME        DEP    SALARY
---------- -------------- --- ----------
    102 mano         12      20000
    105 sree         10      20000
```

SQL> select * from employee where ename in('sree','mano','priya');

```
    ENO ENAME        DEP    SALARY
---------- -------------- --- ----------
    102 mano         12      20000
    103 priya        12      15000
    105 sree         10      20000
```

SQL> select * from employee where ename not in('sree','mano','priya');

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
104	shiva	14	14000

SQL> select * from employee where salary between 12000 and 15000;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15000
103	priya	12	15000
104	shiva	14	14000

SQL> select * from employee where deptid=12 and salary>15000;

ENO	ENAME	DEP	SALARY
102	mano	12	20000

SQL> select * from employee where deptid=12 or salary>15000;

ENO	ENAME	DEP	SALARY
102	mano	12	20000
103	priya	12	15000
105	sree	10	20000

SQL> select * from employee where ename like 's%';

ENO	ENAME	DEP	SALARY
104	shiva	14	14000
105	sree	10	20000

SQL> select * from employee where ename like 's___';

ENO	ENAME	DEP	SALARY
104	shiva	14	14000

RESULT:

The logical operators are executed successfully and thus the expected output is obtained.

Ex.No: 4 Odd or Even Number checking

Code:

```
set serveroutput on

declare
no number;

begin
no:=&no;
if mod(no,2)=0 then
 dbms_output.put_line(no || ' is even');
else
 dbms_output.put_line(no || ' is odd');
end if;
end;
/
```

Output:

```
SQL> /
Enter value for no: 4
old   5: no:=&no;
new   5: no:=4;
4 is even

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:5 SEARCHED CASE

Code:

```
set serveroutput on
declare
no number;
begin
no := &percentage;
case
when no>=80 then dbms_output.put_line('distinction');
when no>=60 and no<80 then dbms_output.put_line('first class');
when no>=50 and no<60 then dbms_output.put_line('Second class');
when no>=35 then dbms_output.put_line('third class');
else
dbms_output.put_line('invalid value');
end case;
end;
/
```

Output:

```
SQL> @d:\dbms\scase.sql
Enter value for percentage: 78
old   5: no := &percentage;
new   5: no := 78;
first class
PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:6 **STRING FUNCTION**

```
SQL> select upper('oracle') from dual;

UPPER(
------
ORACLE


SQL> select lower('ORACLE') "LOWER" from dual;

LOWER
------
oracle


SQL> select initcap('oracle')  "ANSWER" from dual;

ANSWER
------
Oracle


SQL> select concat('R','DBMS') from dual;

CONCA
-----
RDBMS


SQL> select substr('ORACLE',4,2) from dual;

SU
--
CL


SQL> select substr('ORACLE',4) from dual;

SUB
---
CLE


SQL> select instr('ORACLE','E') from dual;

INSTR('ORACLE','E')
-------------------
                  6
```

```
SQL> select ltrim('ORACLE','ORA') from dual;

LTR
---
CLE



SQL> select rtrim('ORACLE','LE') from dual;

RTRI
----
ORAC


SQL> select trim('   ORACLE   ') from dual;

TRIM('
------
ORACLE


SQL> select rtrim('   ORACLE   ') from dual;

RTRIM('OR
---------
   ORACLE


SQL> select ltrim('   ORACLE   ') from dual;

LTRIM('OR
---------
ORACLE


SQL> select trim(leading 'o' from 'oracle') from dual;

TRIM(
-----
racle

SQL> select trim(trailing 'e' from 'oracle') from dual;

TRIM(
-----
oracl
```

```
SQL> select length('oracle') from dual;

LENGTH('ORACLE')
----------------
               6


SQL> select length('  oracle  ') from dual;

LENGTH('ORACLE')
----------------

              10

SQL> select lpad('oracle',10,'*') from dual;

LPAD('ORAC
----------
****oracle


SQL> select rpad('oracle',10,'*') from dual;

RPAD('ORAC
----------
oracle****

SQL> select lpad('ABC',6,'Mr.') from dual;

LPAD('
------
Mr.ABC


SQL> select replace('Gremmer','e','a') from dual;

REPLACE
-------
Grammar
```

RESULT:

The string functions are executed successfully and thus the expected output is obtained.

EX.NO:7 **NUMERIC FUNCTION**

```
SQL> select round(25.465,2) from dual;

ROUND(25.465,2)
---------------
          25.47


SQL> select round(25.465,0) from dual;

ROUND(25.465,0)
---------------
             25

SQL> select trunc(25.465,2) from dual;

TRUNC(25.465,2)
---------------
          25.46


SQL> select trunc(25.655,0) from dual;

TRUNC(25.655,0)
---------------
             25


SQL> select power(2,4) from dual;

POWER(2,4)
----------
        16

SQL> select abs(-23) from dual;

  ABS(-23)
----------
        23

SQL> select mod(23,5) from dual;

 MOD(23,5)
----------
         3

SQL> select sign(-5) from dual;
```

```
  SIGN(-5)
----------
        -1

SQL> select sign(5) from dual;

   SIGN(5)
----------
         1

SQL> select sign(0) from dual;

   SIGN(0)
----------
         0

SQL> select floor(25.6) from dual;

FLOOR(25.6)
-----------
         25

SQL> select ceil(25.4) from dual;

CEIL(25.4)
----------
        26

SQL> select greatest(25,45,34,57) from dual;

GREATEST(25,45,34,57)
---------------------
                   57

SQL> select least(25,45,34,57) from dual;

LEAST(25,45,34,57)
------------------
                25
```

RESULT:

The numeric functions are executed successfully and thus the expected output is obtained.

EX.NO:8 **DATE FUNCTION**

```
select sysdate from dual;

SYSDATE
---------
30-SEP-21


select add_months('25-MAY-21',3) from dual;

ADD_MONTH
---------
25-AUG-21


select next_day('25-may-21', 'WEDNESDAY') from dual;

NEXT_DAY(
---------
26-MAY-21


select last_day('05-DEC-21') from dual;

LAST_DAY(
---------
31-DEC-21


select ROUND(TO_DATE('16-DEC-21'),'month') from dual;

ROUND(TO_
---------
01-JAN-22


select extract(month from sysdate) from dual;

EXTRACT(MONTHFROMSYSDATE)
-------------------------
9
```

```
select months_between('30-sep-21', '12-feb-18') from dual;

MONTHS_BETWEEN('30-SEP-21','12-FEB-18')
---------------------------------------
43.5806452


select new_time('23-jun-21','CST','PST') from dual;

NEW_TIME(
---------
22-JUN-21


select trunc(to_date('01-oct-21'),'day') from dual;

TRUNC(TO_
---------
26-SEP-21


select current_date from dual;

CURRENT_D
---------
30-SEP-21
```

RESULT:

The date functions are executed successfully and thus the expected output is obtained.

EX.NO:9 EMPLOYEE DETAILS (DDL COMMANDS)

Perform the following DDL commands.

1. Create an employee table with the mentioned constraints.

Field name	Constraint
Eid	**Primary Key**
Ename	
Deptid	**Not null**
State	**Default**
Salary	**Check**
Contactno	**Unique**

Create a department table having the fields deptid and deptname and make deptid as primary key.

```
SQL> create table employee(eid number(3) primary key,ename
varchar2(15),deptid number(2) not null, state varchar2(15) default 'TN',
salary number(7) check(salary>=7000),contactno number(10) unique);

Table created.

SQL> create table department(deptid number(2),deptname varchar2(10),
primary key(deptid));

Table created.
```

2. Add a new column to the department table as building_name and update the values.

```
SQL> alter table department add building_name varchar2(15);

Table altered.

SQL> desc department

 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
DEPTID                                    NOT NULL NUMBER(2)
DEPTNAME                                           VARCHAR2(10)
BUILDING_NAME                                      VARCHAR2(15)
```

3. Add foreign key constraint to the deptid column in the employee table.

```
SQL> alter table employee add foreign key(deptid) references
department(deptid);

Table altered.


SQL> desc employee;

 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
EID                                       NOT NULL NUMBER(3)
 ENAME                                              VARCHAR2(15)
 DEPTID                                    NOT NULL NUMBER(2)
 STATE                                              VARCHAR2(15)
 SALARY                                             NUMBER(7)
 CONTACTNO                                          NUMBER(10)
```

4. Remove the not null constraints of deptid column in the employee table.

```
SQL> select constraint_name, column_name from user_cons_columns where
table_name = 'EMPLOYEE';


CONSTRAINT_NAME              COLUMN_NAME
-------------------  ----------------------
SYS_C004072                 CONTACTNO
SYS_C004071                 EID
SYS_C004070                 SALARY
SYS_C004069                 DEPTID
SYS_C004074                 DEPTID


SQL> alter table EMPLOYEE drop constraint SYS_C004069;

Table altered.
```

```
SQL> desc employee

 Name                                      Null?    Type

 ----------------------------------------- -------- ----------------------
EID                                       NOT NULL NUMBER(3)
 ENAME                                              VARCHAR2(15)
 DEPTID                                             NUMBER(2)
 STATE                                              VARCHAR2(15)
 SALARY                                             NUMBER(7)
 CONTACTNO                                          NUMBER(10)
```

5. Change the default value of state column in the employee table.

```
SQL> alter table employee modify state default 'kerala';

Table altered.
```

6. Change eid column data type.

```
SQL> alter table employee modify eid varchar2(5);

Table altered.

SQL> desc employee;

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
EID                                       NOT NULL VARCHAR2(5)
 ENAME                                              VARCHAR2(15)
 DEPTID                                             NUMBER(2)
 STATE                                              VARCHAR2(15)
 SALARY                                             NUMBER(7)
 CONTACTNO                                          NUMBER(10)
```

7. Remove the column state from employee table.

```
SQL> alter table employee drop column state;

Table altered.
```

```
SQL> desc employee

 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
EID                                       NOT NULL VARCHAR2(5)
 ENAME                                              VARCHAR2(15)
 DEPTID                                             NUMBER(2)
 SALARY                                             NUMBER(7)
 CONTACTNO                                          NUMBER(10)
```

8. Rename the eid column as emp_id in the employee table.

```
SQL> alter table employee rename column eid to emp_id;

Table altered.


SQL> desc employee

 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
EMP_ID                                    NOT NULL VARCHAR2(5)
 ENAME                                              VARCHAR2(15)
 DEPTID                                             NUMBER(2)
 SALARY                                             NUMBER(7)
 CONTACTNO                                          NUMBER(10)
```

9. Rename the tables employee and department as emp_details and dept_details respectively.

```
SQL> alter table employee rename to emp_details;
Table altered.


SQL> alter table department rename to dept_details;
Table altered.
```

10. Update the size of Building_name column in the dept_details table.

```
SQL> alter table dept_details modify building_name varchar2(30);

Table altered.
```

11. Remove both the tables.

```
SQL> drop table emp_details;

Table dropped.

SQL> drop table dept_details;

Table dropped.
```

RESULT:

The DDL commands are executed successfully and thus the expected output is obtained.

EX.NO:10 PATIENT DETAILS(DML COMMANDS)

```
SQL> create table patient(pid number, pname varchar(20),age
number,dr_id number);

Table created.

SQL> select * from patient;

       PID PNAME                          AGE      DR_ID
---------- -------------------- ---------- ----------
       123 amal                           17          1
       345 amal                           17          1
       567 nada                           19          2
       891 maha                           20          1
       523 norah                          25          4
        21 maha                           43          6
```

- ✓ present patient age whose id is 567
  select age from patient where pid=567;

- ✓ Present name of patients
  select p_name from patient;

- ✓ Present id of patient whose age is 17 and name amal
  select pid from patient where age=17 and pname='amal';

- ✓ Present id of patient that his name begin with n
  select pid from patient where pname like 'n%';

- ✓ Present phone number of maha
  select mobileno from patient where pname='maha';

- ✓ Present name of patient whose doctor number is 1 or name is amal
  select pname from patient where dr_id=1 or pname='amal';

- ✓ Present different patient name
  select distinct(pname) from patient;

- ✓ Present patients id and their names as one column named patient data
  select pid || pname as patient from patient;

- ✓ Present patients phone number for nada and maha
  select mobileno from patient where pname='nada' or 'maha';

RESULT:

The DML commands are executed successfully and thus the expected output is obtained.

EX.NO:11 VIEW

```
SQL> create table product_details (pid number, pname varchar(20), MRP
number(8),  purchase_rate number(8), sale_rate number(8));

Table created.


SQL> insert into product_details
values(&pid,'&pname',&MRP,&purchase_rate,&sale_rate);
Enter value for pid: 100
Enter value for pname: Pen
Enter value for mrp: 50
Enter value for purchase_rate: 30
Enter value for sale_rate: 45
old   1: insert into product_details
values(&pid,'&pname',&MRP,&purchase_rate,&sale_rate)
new   1: insert into product_details values(100,'Pen',50,30,45)

1 row created.


SQL> select * from product_details;

       PID PNAME                          MRP PURCHASE_RATE  SALE_RATE
---------- -------------------- ---------- ------------- ----------
       100 Pen                            50            30         45
       101 Pencil                         10             4          8
       102 Crayons                        50            35         48


SQL> create view price_details(pid,pname,rate) as select pid,pname,sale_rate from
product_details;

View created.
```

```
SQL> select * from price_details;

       PID PNAME                       RATE
---------- -------------------- ----------
       100 Pen                          45
       101 Pencil                        8
       102 Crayons                      48


SQL> insert into product_details values(103,'Marker',30,20,25);

1 row created.


SQL> select * from price_details;

       PID PNAME                       RATE
---------- -------------------- ----------
       100 Pen                          45
       101 Pencil                        8
       102 Crayons                      48
       103 Marker                       25

SQL> create or replace view price_details as select pid,pname,mrp,sale_rate ra
 from product_details;

View created.

SQL> select * from price_details;

       PID PNAME                           MRP       RATE
---------- ------------------------ ---------- ----------
       100 Pen                              50         45
       101 Pencil                           10          8
       102 Crayons                          50         48

SQL> drop view price_details;

View dropped.
```

RESULT:

The view commands are executed successfully and thus the expected output is obtained.

EX.NO:12 **PRODUCT DETAILS**

(PRIMARY & FOREIN KEY CONSTRAINT)

1. Create the tables.

Table name : Product	
Field name	**Constraint**
pid	Primary Key
pname	Not null
price	

Table name : order_details	
Field name	**Constraint**
Order_id	Primary Key
pid	Foreign Key
Qty	check

```
SQL>  create table product(pid number(2) primary key,pname varchar2(15) not
null,price number(4));

Table created.

SQL>  insert into product values(&pid,'&pname',&price);

SQL> select * from product;

       PID PNAME                PRICE
---------- --------------- ----------
        11 Pen                     40
        12 Pencil                  10
        13 Marker                  80
        14 scale                   10


SQL>  create table order_details(order_id number(2) primary key,pid
number(2),qty number(2) check(qty>30),foreign key(pid) references
product(pid));

Table created.


SQL>  insert into order_details values(&order_id,&pid,&qty);
Enter value for order_id: 1
Enter value for pid: 11
Enter value for qty: 10
old   1:  insert into order_details values(&order_id,&pid,&qty)
new   1:  insert into order_details values(1,11,10)
```

```
 insert into order_details values(1,11,10)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.SYS_C004029) violated


SQL> /
Enter value for order_id: 1
Enter value for pid: 11
Enter value for qty: 40
old   1:  insert into order_details values(&order_id,&pid,&qty)
new   1:  insert into order_details values(1,11,40)

1 row created.


SQL> /
Enter value for order_id: 2
Enter value for pid: 12
Enter value for qty: 50
old   1:  insert into order_details values(&order_id,&pid,&qty)
new   1:  insert into order_details values(2,12,50)

1 row created.

SQL> /
Enter value for order_id: 3
Enter value for pid: 15
Enter value for qty: 40
old   1:  insert into order_details values(&order_id,&pid,&qty)
new   1:  insert into order_details values(3,15,40)
 insert into order_details values(3,15,40)
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.SYS_C004031) violated - parent key
not
found



SQL> select * from order_Details;

  ORDER_ID        PID        QTY
---------- ---------- ----------
         1         11         40
         2         12         50


SQL> delete from product where pid=11;
```

```
delete from product where pid=11
*
ERROR at line 1:
ORA-02292: integrity constraint (SYSTEM.SYS_C004031) violated - child
record
found
```

RESULT:

The commands are executed successfully and thus the expected output is obtained.

Ex.No: 13 Biggest among 3 Nos

Code:

```
set serveroutput on

declare
a number;
b number;
c number;

begin
a:=&a;
b:=&b;
c:=&c;

if(a>b)and (a>c) then
 dbms_output.put_line(a || ' is big ');
else if(b>c) then
 dbms_output.put_line(b || ' is big ');
else
 dbms_output.put_line(c || ' is big ');
end if;
end if;

end;

/
```

Output:

```
SQL> @d:\vk\big.sql
Enter value for a: 12
old   7: a:=&a;
new   7: a:=12;
Enter value for b: 45
old   8: b:=&b;
new   8: b:=45;
Enter value for c: 33
old   9: c:=&c;
new   9: c:=33;
45 is big

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No: 14 Armstrong Number Checking

Code:

```
SET SERVEROUTPUT ON;
DECLARE
num NUMBER;
rev NUMBER;

BEGIN
num := &num;
rev := 0;
WHILE num>0
LOOP
rev:=(rev*10) + mod(num,10);
num:=floor(num/10);
END LOOP;
DBMS_OUTPUT.PUT_LINE('Reverse of the number is: ' || rev);
END;
/
```

Output:

```
SQL> @d:\vk\rev.txt
Enter value for num: 123
old   6: num := &num;
new   6: num := 123;
Reverse of the number is: 321

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No: 15 EMPLOYEE DETAILS (PL/SQL RECORD)

Code:

```
set serveroutput on

DECLARE
   type emp is record
      (e_no  number(3),e_name  varchar(20),
       e_dept number(2),e_salary   number(6));

   emp_rec emp;

BEGIN
 select * into emp_rec from employee where eno=&eno;

   dbms_output.put_line ('ID :  ' || emp_rec.e_no);
   dbms_output.put_line('NAME : ' || emp_rec.e_name);
   dbms_output.put_line('DEPT : ' || emp_rec.e_dept);
   dbms_output.put_line('SALARY : ' || emp_rec.e_salary);
END;
/
```

Output:

```
SQL> @d:\dbms\dbmslab\pl.sql
Enter value for eno: 105
old  11:  select * into emp_rec from employee where eno=&eno;
new  11:  select * into emp_rec from employee where eno=105;
ID :  105
NAME : sree
DEPT : 10
SALARY : 21000

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No: 16 EMPLOYEE DETAILS (PL/SQL TABLE)

Code:

```
set serveroutput on

DECLARE
  TYPE ename_table_type IS TABLE OF employee.ename%TYPE
    INDEX BY BINARY_INTEGER;
  ename_table ename_table_type;
  v_index BINARY_INTEGER:=1;

BEGIN
  FOR emp_record IN (SELECT * FROM employee) LOOP
    ename_table(v_index):=emp_record.ename;
    v_index:=v_index+1;
  END LOOP;
  v_index:=ename_table.FIRST;
  WHILE ename_table.EXISTS(v_index) LOOP
    DBMS_OUTPUT.PUT_LINE(v_index||'  '||ename_table(v_index));
    v_index:=ename_table.NEXT(v_index);
  END LOOP;
END;
/
```

Output:

```
SQL> @d:\dbms\dbmslab\plt.sql
1  ranjani
2  mano
3  priya
4  shiva
5  sree

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No: 17 **JOIN OPERATIONS**

```
create table employee1(eid number,ename varchar(25), dept
varchar(20));
create table emp1_details(id number,salary number);
```

```
SQL> select * from employee1;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1001 shibana                   research
      1002 abhishek                  finance
      1003 pavithra                  finance
      1004 tharini                   transport

SQL> select * from emp1_details;

        ID     SALARY
---------- ----------
      1001      20000
      1003      13000
      1004      16000
```

```
SQL> select * from employee1 inner join emp1_details on
employee1.eid = emp1_details.id;

EID ENAME                 DEPT         ID     SALARY
--- -----    ------------ ---------   ----- ----------
1001 shibana           research       1001      20000
1003 pavithra          finance        1003      13000
1004 tharini           transport      1004      16000

SQL> select * from employee1 left outer join emp1_details on
employee1.eid = emp1_details.id;

EID ENAME                 DEPT         ID     SALARY
--- -----    ------------ ---------   ----- ----------
1001 shibana        research          1001      20000
1003 pavithra       finance           1003      13000
1004 tharini        transport         1004      16000
```

```
1002 abhishek       finance

SQL> select * from employee1 right outer join emp1_details on
employee1.eid = emp1_details.id;

EID ENAME                DEPT         ID     SALARY
--- -----    ------------- ---------   ----- ----------
1001 shibana         research          1001      20000
1003 pavithra        finance           1003      13000
1004 tharini         transport         1004      16000

SQL> select * from employee1 full join emp1_details on
employee1.eid = emp1_details.id;

EID ENAME                DEPT         ID     SALARY
--- -----    ------------- ---------   ----- ----------
1001 shibana         research         1001      20000
1003 pavithra        finance          1003      13000
1004 tharini         transport        1004      16000
1002 abhishek        finance
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No: 18 SET Operations

```
create table employee1(eid number,ename varchar(25), dept
varchar(20));
create table employee2(eid number,ename varchar(25), dept
varchar(20));

insert into employee1 values(&eid, '&ename', '&dept');
insert into employee2 values(&eid, '&ename', '&dept');



SQL> select * from employee1;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1001 shibana                   research
      1002 abhishek                  finance
      1003 pavithra                  finance
      1004 tharini                   transport

SQL> select * from employee2;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1001 shibana                   research
      1006 poorna                    transport
      1007 sree                      finance

SQL> select * from employee1 union select * from employee2;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1001 shibana                   research
      1002 abhishek                  finance
      1003 pavithra                  finance
      1004 tharini                   transport
      1006 poorna                    transport
      1007 sree                      finance

6 rows selected.
```

```
SQL> select * from employee1 union all select * from employee2;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1001 shibana                   research
      1002 abhishek                  finance
      1003 pavithra                  finance
      1004 tharini                   transport
      1001 shibana                   research
      1006 poorna                    transport
      1007 sree                      finance

7 rows selected.

SQL> select * from employee1 intersect select * from employee2;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1001 shibana                   research

SQL> select * from employee1 minus select * from employee2;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1002 abhishek                  finance
      1003 pavithra                  finance
      1004 tharini                   transport

SQL> select * from employee2 minus select * from employee1;

       EID ENAME                     DEPT
---------- ------------------------- --------------------
      1006 poorna                    transport
      1007 sree                      finance
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:19 FACTORIAL OF A NUMBER

Function Code:

```
SQL> edit d:\vk\fact.sql

create or replace function fact(n number)return number is
i integer;
f integer;
begin
f:=1;
for i in 1..n
loop
f:=f*i;
end loop;
return(f);
end;
/
```

Output:

```
SQL> @d:\vk\fact.sql
Function created.
```

Program Code:

```
SQL> edit d:\vk\fact_pgm.sql
set serveroutput on

declare
n number;
begin
n:=&n;
dbms_output.put_line('The result is ' || fact(n));
end;
/
```

Output:

```
SQL> @d:\vk\fact_pgm.sql
Enter value for n: 3
old   4: n:=&n;
new   4: n:=3;
The result is 6

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:20 **Palindrome checking (String)**

Function Code:

```
create or replace function palindrome(s string)return varchar is
r varchar(30);
c varchar(30);
v varchar(30);

begin
v := lower(s);
for i in reverse 1..length(v)
loop
c := substr(v,i,1);
r := r||c;
end loop;

if(v=r) then
return('Palindrome');
else
return('Not Palindrome');
end if;
end;
/
```

Output:

```
SQL> @d:\dbms\dbmslab\pal.sql
Function created.
```

Program Code:

```
set serveroutput on

declare
s varchar(20);
r varchar(20);
begin
s := '&s';
r := palindrome(s);
dbms_output.put_line(r);
end;

/
```

Output:

```
SQL> @d:\dbms\dbmslab\pal_code.sql
Enter value for s: Malayalam
old   5: s := '&s';
new   5: s := 'Malayalam';
Palindrome

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:21 **Square of number**

Procedure Code:

```
create or replace procedure squareval(x in out number) is
begin
x :=x*x;
end;
/
```

Output:

```
SQL> @c:\users\vk\sqr_proc.txt
Procedure created.
```

Program Code:

```
set serveroutput on

declare
a number;

begin
a := &a;
squareval(a);
dbms_output.put_line('The Squared value is  ' ||a);
end;
/
```

Output:

```
SQL> @c:\users\vk\sqr.txt
Enter value for a: 12
old   5: a := &a;
new   5: a := 12;
The Squared value is  144

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:22 **Sum of 2 values**

Procedure Code:

```
create or replace procedure sum(x in number, y in out number) is
begin
 y := x+y;
end;
/
```

Output:

```
SQL> @c:\users\vk\sum_proc.txt

Procedure created.
```

Program Code:

```
declare
a number;
b number;
begin
a:=&a;
b:=&b;
sum(a,b);
dbms_output.put_line('Sum is ' || b);
end;
/
```

Output:

```
SQL> @c:\users\vk\sum.txt
Enter value for a: 12
old   5: a:=&a;
new   5: a:=12;
Enter value for b: 14
old   6: b:=&b;
new   6: b:=14;
Sum is 26

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:23 Inserting records

Table:

```
SQL> create table student_details(rno number, name varchar(25), dept_name
varchar(10));

Table created.
```

Procedure Code:

```
create or replace procedure add_row(rno in number, name in varchar,
dept_name in varchar) is
begin
insert into student_details values(rno,name,dept_name);
dbms_output.put_line('Record inserted successfully');
end;
/
```

Output:

```
SQL> @c:\users\vk\insert_pro.txt

Procedure created.
```

Program Code:

```
declare
rno number;
name varchar(25);
dept_name varchar(10);

begin
rno := &rno;
name := '&name';
dept_name := '&dept_name';
add_row(rno,name,dept_name);
end;
/
```

Output:

```
SQL> @c:\users\vk\insert.txt
Enter value for rno: 501
old   7: rno := &rno;
new   7: rno := 501;
Enter value for name: santhosh
old   8: name := '&name';
new   8: name := 'santhosh';
Enter value for dept_name: CSE
old   9: dept_name := '&dept_name';
new   9: dept_name := 'CSE';
Record inserted successfully

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:24 No_Data_Found (Predefined Exception)

Table:

```
SQL> create table product(pid number primary key, pname varchar2(20));

Table created.


SQL> select * from product;

       PID PNAME
---------- --------------------
       101 Pen
       102 Pencil
       103 Marker
       104 Acrylic paint
       105 crayons
       106 color pencil

6 rows selected.
```

Code:

```
declare
id product.pid%type;
name product.pname%type;

begin
id := &id;
select pid,pname into id,name from product where pid = id ;
dbms_output.put_line(id|| ' '||name);
exception
when no_data_found then
dbms_output.put_line('No such record');
end;
/
```

Output:

```
SQL> @c:\users\vk\ex1.txt
Enter value for id: 102
old   6: id := &id;
new   6: id := 102;
102 Pencil

PL/SQL procedure successfully completed.

SQL> /
Enter value for id: 110
old   6: id := &id;
new   6: id := 110;
No such record

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:25 Zero_divide (Predefined Exception)

Code:

```
set serveroutput on

declare
a number;
b number;
begin
a:=&a;
b:=&b;
a:= a/b;
dbms_output.put_line('Result : ' || a);
exception
when zero_divide then
dbms_output.put_line('Divide by zero  is not allowed');
end;
/
```

Output:

```
SQL> @d:\vk\zero.txt
Enter value for a: 12
old   6: a:=&a;
new   6: a:=12;
Enter value for b: 3
old   7: b:=&b;
new   7: b:=3;
Result : 4

PL/SQL procedure successfully completed.

SQL> /
Enter value for a: 12
old   6: a:=&a;
new   6: a:=12;
Enter value for b: 0
old   7: b:=&b;
new   7: b:=0;
Divide by zero  is not allowed
PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:26 User defined Exception

Code:

```
set serveroutput on

declare
a number;
b number;
divide_by_zero exception;

begin
a:=&a;
b:=&b;
if(b=0) then
raise divide_by_zero;
else
dbms_output.put_line('Result ' || a/b);
end if;
exception
when divide_by_zero then
dbms_output.put_line('Divide by zero  is not allowed');
end;

/
```

Output:

```
SQL> @d:\vk\userex.txt
Enter value for a: 10
old   7: a:=&a;
new   7: a:=10;
Enter value for b: 2
old   8: b:=&b;
new   8: b:=2;
Result 5

PL/SQL procedure successfully completed.
```

```
SQL> /
Enter value for a: 10
old   7: a:=&a;
new   7: a:=10;
Enter value for b: 0
old   8: b:=&b;
new   8: b:=0;
Divide by zero  is not allowed

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:27 IMPLICIT CURSOR

SQL> select * from employee;

ENO	ENAME	DEP	SALARY
101	ranjani	14	15500
102	mano	12	20500
103	priya	12	15500
104	shiva	14	14500
105	sree	10	20500

Code:

```
DECLARE
  total_rows number(2);
BEGIN
  UPDATE employee SET salary = salary + 500;
  IF sql%notfound THEN
    dbms_output.put_line('no employee selected');
  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( 'salary for ' || total_rows || ' is/are updated ');
  END IF;
END;
/
```

Output:

```
SQL> @d:\dbms\dbmslab\cursor_attributes.sql
salary for 5 is/are updated

PL/SQL procedure successfully completed.

SQL> select * from employee;


       ENO ENAME           DEP     SALARY
---------- --------------- --- ----------
       101 ranjani         14       16000
       102 mano            12       21000
       103 priya           12       16000
       104 shiva           14       15000
       105 sree            10       21000
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:28 EXPLICIT CURSOR USING BASIC LOOP

Code:

```
DECLARE
CURSOR stud IS SELECT * FROM student;
v_sname student.sname%type;
v_rno student.rno%type;
v_dept student.dept%type;

BEGIN
OPEN stud;

LOOP
FETCH stud INTO v_rno,v_sname,v_dept;
IF stud%NOTFOUND
THEN
EXIT;
END IF;
Dbms_output.put_line(v_rno|| '  ' || v_sname || '  '||v_dept);
END LOOP;

Dbms_output.put_line('Total no.of records fetched : '  || stud%rowcount);
CLOSE stud;
END;
/
```

Output:

SQL> @d:\dbms\dbmslab\loop_cursor.sql

101 kanishka cs

102 aarika tam

103 thasmitha eng

104 sakthi bala cs

Total no.of records fetched : 4

PL/SQL procedure successfully completed.

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

EX.NO:29 EXPLICIT CURSOR USING FOR LOOP

Code:

```
DECLARE
CURSOR stud_cur IS SELECT * FROM student;

BEGIN

FOR stud IN stud_cur
LOOP
dbms_output.put_line(stud.rno || ' '|| stud.sname ||' ' || stud.dept);
END LOOP;

END;
/
```

Output:

```
SQL> @d:\dbms\dbmslab\forcursor.sql
101 kanishka cs
102 aarika tam
103 thasmitha eng
104 sakthi bala cs

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No:30 Age checking (Trigger)

Table:

```
create table voterid_applied(app_no number,name varchar(20),age
number,place varchar(20));

Table created
```

Trigger Code:

```
create or replace trigger checkage before insert or update on
voterid_applied for each row
begin
if :new.age<18 then
raise_application_error(-20001,'Age should be equal to 18 or greater
than 18');
end if;
end;
/
```

Output:

```
SQL> @d:\vk\age_trig.sql
Trigger created.


SQL> insert into voterid_applied
values(&app_no,'&name',&age,'&place');
Enter value for app_no: 1432
Enter value for name: Vinitha
Enter value for age: 18
Enter value for place: Cumbum
old   1: insert into voterid_applied
values(&app_no,'&name',&age,'&place')
new   1: insert into voterid_applied
values(1432,'Vinitha',18,'Cumbum')

1 row created.
```

```
SQL> /
Enter value for app_no: 1433
Enter value for name: Anitha
Enter value for age: 17
Enter value for place: Madurai
old   1: insert into voterid_applied
values(&app_no,'&name',&age,'&place')
new   1: insert into voterid_applied
values(1433,'Anitha',17,'Madurai')
insert into voterid_applied values(1433,'Anitha',17,'Madurai')
            *
ERROR at line 1:
ORA-20001: Age should be equal to 18 or greater than 18
ORA-06512: at "SYSTEM.CHECKAGE", line 3
ORA-04088: error during execution of trigger 'SYSTEM.CHECKAGE'
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No:31 CASE CONVERSION (TRIGGER)

TABLE CREATION

```
SQL> create table app_received(app_no number(5) primary key, name
varchar2(20), age number(2), place varchar2(15));

Table created.
```

TRIGGER CODE:

```
create or replace trigger change_case before insert or update on
app_received for each row
begin
:new.name := upper(:new.name);
end;
/
```

OUTPUT:

```
SQL> @c:\users\vk\trigger.txt

Trigger created.

SQL> insert into app_received values(&app_no,'&name',&age,'&place');
Enter value for app_no: 1241
Enter value for name: dharshey
Enter value for age: 18
Enter value for place: Thirumangalam
old   1: insert into app_received values(&app_no,'&name',&age,'&place')
new   1: insert into app_received
values(1241,'dharshey',18,'Thirumangalam')

1 row created.


SQL> select * from app_received;

    APP_NO NAME                           AGE PLACE

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

      1241 DHARSHEY                      18 Thirumangalam
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.

Ex.No:32 Arithmetic operations using package

Package Header

```
create or replace package pkg is
function add(x number,y number)return number;
function sub(x number,y number)return number;
procedure mul(x in number,y in number,z out number);
procedure div(x in number,y in number,z out number);
end pkg;
/

SQL> @d:\vk\package.sql

Package created.
```

Package Body

```
create or replace package body pkg is
function add(x number,y number)return number is
z number;
begin
z:=x+y;
return z;
end add;
function sub(x number,y number)return number is
z number;
begin
z:=x-y;
return z;
end sub;
procedure mul(x in number,y in number,z out number)is
begin
z:=x*y;
end mul;
procedure div(x in number,y in number,z out number)is
begin
z:=x/y;
end div;
end pkg;
/
```

OUTPUT:

```
SQL> @d:\vk\package_body.sql

Package body created.
```

Program Code:

```
set serveroutput on;



declare
x number;
y number;
z number;
op char(1);

begin
x:=&x;
y:=&y;
op:='&op';

if(op='+') then
z:=pkg.add(x,y);
dbms_output.put_line('the sum is '||z);
else if(op='-') then
z:=pkg.sub(x,y);
dbms_output.put_line('the different is '||z);
else if(op='*') then
pkg.mul(x,y,z);
dbms_output.put_line('the multiplication is '||z);
else if(op='/') then
pkg.div(x,y,z);
dbms_output.put_line('the division is '||z);
end if;
end if;
end if;
end if;
end;
/
```

OUTPUT:

```
SQL> @d:\vk\arith.sql
Enter value for x: 14
old   8: x:=&x;
new   8: x:=14;
Enter value for y: 2
old   9: y:=&y;
new   9: y:=2;
Enter value for op: /
old  10: op:='&op';
new  10: op:='/';
the division is 7

PL/SQL procedure successfully completed.
```

RESULT:

The PL/SQL program is executed successfully and thus the expected output is obtained.