PROGRAMS ON PL/SQL:

1. a. Write a PL/SQL program to swap two numbers.

   b. Write a PL/SQL program to find the largest of three numbers.

Ans.

a.

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

```
end;
/
```

b.

```
declare
a number(3);
b number(3);
begin
        a:=&a;
        b:=&b;
        dbms_output.put_line('Before swapping a='||a||' and b='||b||');
        a:=a+b;
        b:=a-b;
        a:=a-b;
        dbms_output.put_line('After swapping a='||a||' and b='||b||');
        end;
/
```

2.

a. Write a PL/SQL program to find the total and average of 4 subjects and display the grade.

b. Write a program to accept a number and find the sum of the digits.

Ans.

a.

```
declare
java number(10);
dbms number(10);
co number(10);
mfcs number(10);
total number(10);
avgs number(10);
per number(10);
begin
        dbms_output.put_line('ENTER THE MARKS');
         java:=&java;
        dbms:=&dbms;
         co:=&co;
          mfcs:=&mfcsl;
         total:=(java+dbms+co+mfcs);
         per:=(total/600)*100;
         if java<40 or dbms<40 or co<40 or mfcs<40 then
```

```
        dbms_output.put_line('FAIL');
    if per>75 then
        dbms_output.put_line('GRADE A');
    elsif per>65 and per<75 then
        dbms_output.put_line('GRADE B');
    elsif per>55 and per<65 then
        dbms_output.put_line('GRADE C');
    else
        dbms_output.put_line('INVALID INPUT');
    end if;
        dbms_output.put_line('PERCENTAGE IS '||per);
end;
/
```

b.

```
declare
n number(5):=&n;
s number:=0;
r number(2):=0;
begin
    while n !=0
```

```
        loop
        r:=mod(n,10);
        s:=s+r;
        n:=trunc(n/10);
        end loop;
dbms_output.put_line('sum of digits of given number is '||s);
end;
/
```

3.

a. PL/SQL Program to accept a number from user and print number in reverse order.

b. Write a PL / SQL program to check whether the given number is prime or not.

Ans.

a.

```
declare
num1 number(5);
num2 number(5);
rev number(5);
```

```
begin
        num1:=&num1;
        rev:=0;
        while num1>0
        loop
                num2:=num1 mod 10;
                rev:=num2+(rev*10);
                num1:=floor(num1/10);
        end loop;
        dbms_output.put_line('Reverse number is: '||rev);
end;
/
```

b.

```
declare
        num number;
        i number:=1;
        c number:=0;
begin
        num:=&num;
        for i in 1..num
```

```
        loop
                if((mod(num,i))=0)
                then
                c:=c+1;
                end if;
        end loop;
        if(c>2)
                then
                dbms_output.put_line(num||' not a prime');
        else
                dbms_output.put_line(num||' is prime');
        end if;
end;
/
```

4.

a. Write a PL/SQL program to find the factorial of a given number.

b. calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in table areas. Consisting of two columns radius and area.

Ans.

a.

```
declare
i number(4):=1;
n number(4):=&n;
f number(4):=1;
begin
        for i in 1..n
        loop
                f:=f*i;
        end loop;
        dbms_output.put_line('the factorial of '||n||' is:'||f);
end;
/
```

b.

```
declare
 pi constant number(4,2) := 3.14;
radius number(5);
area number(14,2);
Begin
```

```
    radius := 3;
    While radius <=7
    Loop
        area := pi* power(radius,2);
        Insert into areas values (radius, area);
        radius:= radius+1;
    end loop;
end;
/
```

5.

a. Write a PL/SQL program to accept a string and remove the vowels from the string. (When ‘hello’ passed to the program it should display ‘Hll’ removing e and o from the world Hello).

b. Write a PL/SQL program to accept a number and a divisor. Make sure the divisor is less than or equal to 10. Else display an error message. Otherwise Display the remainder.

Ans.

a.

```
set serveroutput on
set verify off
accept vstring prompt "Please enter your string: ";
```

```
declare
        vnewstring varchar2(100);
begin
        vnewstring := regexp_replace('&vstring', '[aeiouAEIOU]','');
        dbms_output.put_line('The new string is: ' || vnewstring);
end;
/
```

b.

```
select remainder(37,5) "remainder" from dual ;
```

Procedures and Functions:

1. Write a function to accept employee number as parameter and return Basic +HRA together as single column.

Ans.

```
declare
    ename varchar2(15);
    basic number;
    da number;
    hra number;
    pf number;
    netsalary number;
    yearsalary number;
```

```
    begin
        ename:='&ename';
        basic:=&basic;
        da:=basic * (30/100);
        hra:=basic * (10/100);
        if (basic < 8000)
        then
            pf:=basic * (8/100);
        elsif (basic >= 8000 and basic <= 16000)
        then
                pf:=basic * (10/100);
        end if;
        netsalary:=basic + da + hra - pf;
        yearsalary := netsalary*12;
        dbms_output.put_line('Employee name : ' || ename);
        dbms_output.put_line('Providend Fund : ' || pf);
        dbms_output.put_line('Net salary : ' || netsalary);
        dbms_output.put_line('Year salary : '|| yearsalary);
end;
/
```

Output:

Employee name : name
providend fund: 240
net salary : 3960

2. Accept year as parameter and write a Function to return the total net salary spent for a given year.

Ans.

```
SQL> select * from works;
 EMP_NO      COMPANY NAME  JOINING_D  DESIGNATION    SALARY
DEPTNO
 1 abc 23-NOV-00 project lead 40000 1
2 abc 25-DEC-10 software engg 20000 2
3 abc 15-JAN-11 software engg 1900 1
4 abc 19-JAN-11 software engg 19000 2
5 abc 06-FEB-11 software engg 18000 1
SQL> get e:/plsgl/p15.sgl;
1 Create or replace function tot_sal_of_dept(dno number)
2 return number
3 is
4 tot sal number:-0;
5 begin
6 select sum(salary) into tot_sal from works where deptno-dno;
7 return tot_sal;
8* end;
 SQL> .
 SQL> /
Function created.

SQL> begin
 2 dbms_output.put_linerTotal salary of DeptNo 1 is :' ||
tot_sal_of_dept (1) I ;
 3 end;
 4 .
```

```
SQL> set serveroutput on;
SQL> /
Total salary of DeptNo 1 is :77000
PL/SQL procedure successfully completed.
```

3. Create a function to find the factorial of a given number and hence find NCR.

Ans.

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

```
/
SQL> create or replace function ncr(n number ,r number)
return number is  n1 number:=fact(n);
r1 number:=fact(r);
nr1 number:=fact(n-r);
result number;
begin
        result:=(n1)/(r1*nr1);
        return result;
end;
/
```

4. Write a PL/SQL block to print Fibonacci series using local functions.

Ans.

```
>create or replace function fib (n positive) return integer is
begin
    if (n = 1) or (n = 2) then -- terminating condition
    return 1;
    else
    return fib(n - 1) + fib(n - 2); -- recursive call
```

```
        end if;
end fib;
/
```

5. Create a procedure to find the lucky number of a given birth date.

Ans.

```
SQL>set serverout on
SQL>declare
  l_input varchar2(20) := '31/01/1978';
l_output int;
begin
    loop
        dbms_output.put_line('------------------');
        dbms_output.put_line('l_input='||l_input);
        l_output := 0;
        for i in 1 .. length(l_input)
        loop
            if substr(l_input,i,1) between '0' and '9' then
                l_output := l_output + to_number(substr(l_input,i,1));
            end if;
            end loop;
```

```
            dbms_output.put_line('l_output='||l_output);
            exit when l_output < 10;
            l_input := to_char(l_output);
            end loop;
            dbms_output.put_line('------------------');
            dbms_output.put_line('Lucky='||l_output);
end;
/
```

Output:

```
------------------
l_input=31/01/1978
l_output=30
------------------
l_input=30
l_output=3
------------------
Lucky=3


PL/SQL procedure successfully completed.
```

6. Create function to the reverse of given number.

Ans.

```
num
Initialize rev_num = 0
Loop while num > 0
   Multiply rev_num by 10 and add remainder of num
      divide by 10 to rev_num
         rev_num = rev_num*10 + num%10;
   Divide num by 10
Return rev_num
```

Accept year as parameter and write a Function to return the total net salary spent for a
given year.

Triggers:

1. Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

   Customer's table:

ID	Name	Age	Address	Salary
1	Alive	24	Khammam	2000
2	Bob	27	Kadappa	3000
3	Catri	25	Guntur	4000
4	Dena	28	Hyderabad	5000
5	Eeshwar	27	Kurnool	6000
6	Farooq	28	Nellore	7000

Ans.

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.salary  - :OLD.salary;
    dbms_output.put_line('Old salary: ' || :OLD.salary);
    dbms_output.put_line('New salary: ' || :NEW.salary);
    dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Trigger created.

2. Convert employee name into uppercase whenever an employee record is inserted or updated. Trigger to fire before the insert or update.

    Ans.

```
SQL> create table Employee(
```

```
ID              VARCHAR2(4 BYTE) NOT NULL,
First_Name      VARCHAR2(10 BYTE),
Last_Name       VARCHAR2(10 BYTE),
Start_Date      DATE,
End_Date        DATE,
Salary          NUMBER(8,2),
City            VARCHAR2(10 BYTE),
Description     VARCHAR2(15 BYTE)
)
/
Table created.
SQL> CREATE OR REPLACE TRIGGER employee_insert_update
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW
DECLARE
        dup_flag  INTEGER;
        BEGIN
                --Force all employee names to uppercase.
                :NEW.first_name := UPPER(:NEW.first_name);
END;
```

/

Trigger created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values ('01','Jason', 'Martin', to_date('19960725','YYYYMMDD'), to_date('20060725','YYYYMMDD'), 1234.56, 'Toronto', 'Programmer')

/

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values('02','Alison', 'Mathews', to_date('19760321','YYYYMMDD'), to_date('19860221','YYYYMMDD'), 6661.78, 'Vancouver','Tester')

/

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values('03','James', 'Smith', to_date('19781212','YYYYMMDD'), to_date('19900315','YYYYMMDD'), 6544.78, 'Vancouver','Tester')

/

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values('04','Celia', 'Rice', to_date('19821024','YYYYMMDD'), to_date('19990421','YYYYMMDD'), 2344.78, 'Vancouver','Manager')

/

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values('05','Robert', 'Black', to_date('19840115','YYYYMMDD'), to_date('19980808','YYYYMMDD'), 2334.78, 'Vancouver','Tester')

/

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values('06','Linda', 'Green', to_date('19870730','YYYYMMDD'), to_date('19960104','YYYYMMDD'), 4322.78,'New York', 'Tester')

/

1 row created.

SQL> insert into Employee(ID, First_Name, Last_Name, Start_Date, End_Date, Salary, City, Description)

values('07','David', 'Larry', to_date('19901231','YYYYMMDD'), to_date('19980212','YYYYMMDD'), 7897.78,'New York', 'Manager')

/

1 row created.

```
SQL> insert into Employee(ID,  First_Name, Last_Name, Start_Date,
End_Date, Salary, City,        Description)


values('08','James',    'Cat',     to_date('19960917','YYYYMMDD'),
to_date('20020415','YYYYMMDD'), 1232.78,'Vancouver', 'Tester')


/

 1 row created.


SQL> select * from Employee

/

ID   FIRST_NAME LAST_NAME  START_DAT END_DATE      SALARY CITY
DESCRIPTION ---- ---------- ---------- --------- --------- ---------- ---------
--------------------------------------------------

01   JASON      Martin      25-JUL-96  25-JUL-06    1234.56  Toronto
Programmer

02   ALISON     Mathews     21-MAR-76  21-FEB-86    6661.78  Vancouver
Tester

03   JAMES      Smith       12-DEC-78  15-MAR-90 6544.78  Vancouver
Tester
```

04 CELIA Rice 24-OCT-82 21-APR-99 2344.78 Vancouver Manager

05 ROBERT Black 15-JAN-84 08-AUG-98 2334.78 Vancouver Tester

06 LINDA Green 30-JUL-87 04-JAN-96 4322.78 New York Tester

07 DAVID Larry 31-DEC-90 12-FEB-98 7897.78 New York Manager

08 JAMES Cat 17-SEP-96 15-APR-02 1232.78 Vancouver Tester

8 rows selected.

SQL> drop table Employee

/

Table dropped.

3. Trigger before deleting a record from emp table. Trigger will insert the row to be deleted into another table and also record the user who has deleted the record.

Ans.

CREATE OR REPLACE TRIGGER employee_before_delete

```
BEFORE DELETE
   ON employee
   FOR EACH ROW
DECLARE
   v_username varchar2(10);
BEGIN
    -- Find username of person performing the DELETE on the table
   SELECT user INTO v_username
    FROM dual;
    -- Insert record into audit table
    INSERT INTO employee_audit (id, salary, delete_date,deleted_by
   )
   VALUES (:old.id,:old.salary, sysdate, v_username );
   END;
   /
Trigger created.
SQL> delete from employee;
8 rows deleted.
SQL> select * from employee_audit;
```

ID SALARY DELETE_DA DELETED_BY ---- ---------- --------- --------------

01 1234.56 09-SEP-06 JAVA2S

02 6661.78 09-SEP-06 JAVA2S

03 6544.78 09-SEP-06 JAVA2S

04 2344.78 09-SEP-06 JAVA2S

05 2334.78 09-SEP-06 JAVA2S

06 4322.78 09-SEP-06 JAVA2S

07 7897.78 09-SEP-06 JAVA2S

08 1232.78 09-SEP-06 JAVA2S

8 rows selected.

SQL> drop table employee_audit;

Table dropped.

Procedures:

1. Create the procedure for palindrome of given number.

Ans.

```
declare

-- declare variable n, m, temp
-- and temp of datatype number
   n number;
   m number;
   temp number:=0;
   rem number;
```

```
begin
   n:=5432112345;
   m:=n;

   -- while loop with condition till n>0
   while n>0
   loop
      rem:=mod(n,10);
      temp:=(temp*10)+rem;
      n:=trunc(n/10);
   end loop; -- end of while loop here

   if m = temp
   then
      dbms_output.put_line('true');
   else
      dbms_output.put_line('false');
   end if;
end;
/
```

2. Write the PL/SQL programs to create the procedure for factorial of given number

Ans.

```
declare
        n number;
        fac number:=1;
        i number;

begin
        n:=&n;
```

```
        for i in 1..n
        loop
                fac:=fac*i;
        end loop;

        dbms_output.put_line('factorial='||fac);
end;
/
```

3. Write the PL/SQL programs to create the procedure to find sum of N natural number.

Ans.

```
Declare
 i number:=0;
 n number;
 sum1 number:=0;
 Begin
 n:=&n;
 while i
 loop
 sum1:=sum1+i;
 dbms_output.put_line(i);
 i:=i+1;
 end loop;
 dbms_output.put_line('The sum is:'||sum1);
 End;
/
```

4. Write the PL/SQL programs to create the procedure to find Fibonacci series.

```
declare
        first number:=0;
```

```
        second number:=1;
        third number;
        n number:=&n;
        i number;

begin
        dbms_output.put_line('Fibonacci series is:');
        dbms_output.put_line(first);
        dbms_output.put_line(second);

        for i in 2..n
        loop
                third:=first+second;
                first:=second;
                second:=third;
                dbms_output.put_line(third);
        end loop;
end;
/
```

5. Write the PL/SQL programs to create the procedure to check the given number is perfect or not.

Ans.

```
declare
n number;
i number;
tot number;
begin
n:=&n;
tot:=0;
for i in 1..n/2
loop
```

```
if(n mod i=0) then
tot:= tot+i;
end if;
end loop;
if(n=tot)then
dbms_output.put_line('Perfect no');
else
dbms_output.put_line('Not a Perfect no');
end if;
end;

/
```

Cursors:

1. Write a PL/SQL block that will display the name, dept no, salary of fist highest paid employees.

Ans.

```
DECLARE
   CURSOR dpt_cur IS
    SELECT d.department_id      id,
          department_name      dptname,
          Nvl(first_name, '...') manager
    FROM   departments d
          left outer join employees e
                    ON ( d.manager_id = e.employee_id )
    ORDER  BY 2;
   emp_name       employees.first_name%TYPE;
   emp_max_salary employees.salary%TYPE;
BEGIN
   FOR dept_all IN dpt_cur LOOP
      SELECT Max(salary)
```

```
 INTO   emp_max_salary
       FROM   employees
       WHERE  department_id = dept_all.id;

       IF emp_max_salary IS NULL THEN
         emp_name := '...';
       ELSE
         SELECT first_name
         INTO   emp_name
         FROM   employees
         WHERE  department_id = dept_all.id
              AND salary = emp_max_salary;
       END IF;

       dbms_output.Put_line(Rpad(dept_all.dptname, 20)
                         || Rpad(dept_all.manager, 15)
          || Rpad(emp_name, 20));
    END LOOP;
END;
/
```

2. Write a PL/SQL block that will display the employee details along with salary using cursors.

```
declare
  cursor c_emp is
    select ename,sal from emp;
  v_name emp.ename%type;
  v_sal emp.sal%type;
begin
  open c_emp;
```

```
  fetch c_emp into v_name,v_sal;
  dbms_output.put_line(v_name||' '||v_sal);
end;
/
```

3. To write a Cursor to find employee with given job and deptno.

Ans.

```
DECLARECURSOR A ISSELECT EMP_NAME,SALARY FROM EMPLOYEE
WHERE DEPTNO = &DEPTNO;
NAME EMPLOYEE.EMP_NAME%TYPE;
SAL EMPLOYEE.SALARY%TYPE;
BEGINOPEN A;
IF A%ISOPEN THEN
 DBMS_OUTPUT.PUT_LINE('DESIRABLE CURSOR HAS BEEN OPENED');
 LOOP
 FETCH A INTO NAME,SAL;
 EXITWHEN A%NOTFOUND;
 DBMS_OUTPUT.PUT_LINE(NAME||'   '||SAL);
 END LOOP;
ELSE
 DBMS_OUTPUT.PUT_LINE('UNABLE TO OPEN THE CURSOR');
ENDIF;
END;
/
```

4. Write a PL/SQL block using implicit cursor that will display message, the salaries of all the employees in the „employee‟ table are updated.If none of the employee‟s salary are updated we get a message 'None of the salaries were updated'. Else we get a message like for example, 'Salaries for

1000 employees are updated’ if there are 1000 rows in „employee‟ table.

Ans.

```
DECLARE  var_rows number(5);
BEGIN
  UPDATE employee
  SET salary = salary + 1000;
  IF SQL%NOTFOUND THEN
    dbms_output.put_line('None of the salaries where updated');
  ELSIF SQL%FOUND THEN
    var_rows := SQL%ROWCOUNT;
    dbms_output.put_line('Salaries for ' || var_rows || 'employees are
updated');
  END IF;
END;
```