Experiment No: 3

SUM OF TWO NUMBERS

AIM

To write a PL/SQL program to print the sum of 2 numbers

ALGORITHM

```
1. Start
2. Set server output on
3- Declare variables a,b,c
4. Set c = a+b
5. Print c
6. Stop
```

RESULT

The program was executed successfully and output was verified.

[illegible]

```
/**************************************************************************/
/*      NAME     : SUSAN SHIBU                                            */
/*      CLASS    : S5 CSE-A                                               */
/*      ROLL NO  : 58                                                     */
/*      DATE     : 06-12-2021                                             */
/*                                                                        */
/*      SUM OF TWO NUMBERS                                                */
/**************************************************************************/
```

PROGRAM :

```
SQL> set serveroutput on;
SQL> declare
  2  a integer:=10;
  3  b integer:=20;
  4  c integer;
  5  begin
  6  c:=a+b;
  7  dbms_output.put_line('sum = '||c);
  8  end;
  9  /
```

OUTPUT:

```
sum = 30

PL/SQL procedure successfully completed.
```

Experiment No: 4

SUM OF FIRST N ODD NUMBERS

AIM

To write a PL/SQL program to print the sum of first n odd numbers

ALGORITHM

```
1. Start
2. Set server output on
3. Declare variables n, s, i, c
4. Set s, i, c to 0
5. while (c > n)
    5.1 If (mod (i,2)!=0)
        5.1.1 Increment s by i
        5.1.2 Increment c by 1
    5.2 Increment i by 1
6. End while
7. Print s
8. Stop.
```

RESULT

The program executed successfully and output was verified.

```
/*************************************************************************/
/*       NAME     : SUSAN SHIBU                                          */
/*       CLASS    : S5 CSE-A                                             */
/*       ROLL NO  : 58                                                   */
/*       DATE     : 10-12-2021                                           */
/*                                                                       */
/*       SUM OF N ODD NUMBERS                                            */
/*************************************************************************/

SQL> set serveroutput on;
SQL>  declare
  2     x integer;
  3    c integer;
  4     sum1 integer;
  5    l integer;
  6    begin
  7   sum1:=0;
  8   x:=1;
  9   c:=&c;
 10   l:=2*c;
 11    while x<l LOOP
 12   sum1:=sum1+x;
 13   x:=x+2;
 14   end loop;
 15   dbms_output.put_line(sum1);
 16   end;
 17   /
Enter value for c: 3
old   9: c:=&c;
new   9: c:=3;
9
```

Experiment No: 5

LARGEST AND SMALLEST OF 3 NUMBERS

AIM

To write a PLSQL program to find the largest and smallest of 3 numbers.

ALGORITHM

```
1. Start
2. Set server output on
3. Declare and read variables a, b, c
4. If (a>b)
    4.1 If (a>c)
        4.1.1 Print a is greater
    4.2 Else
        4.2.1 Print c is greater
5. Else
    5.1 If (b>c)
        5.1.1 Print b is greater.
    5.2 Else
        5.2.1 Print c is greater
6 If (a<b)
    6.1 If (b<c)
        6.1.1 Print a is smaller
    6.2 Else
        6.2.1 Print c is smaller
```

```
/*************************************************************************/
/*       NAME      : SUSAN SHIBU                                        */
/*       CLASS     : S5 CSE-A                                           */
/*       ROLL NO   : 58                                                 */
/*       DATE      : 20-12-2021                                         */
/*                                                                      */
/*               LARGEST AND SMALLEST OF 3 NUMBERS                      */
/*************************************************************************/

SQL> set serveroutput on;
SQL> create or replace procedure p2(a in number,b in number,c in number)
  2  as
  3  l number;
  4  s number;
  5  begin
  6  if a>b then
  7  if a>c then
  8  dbms_output.put_line('largest number is:'||a);
  9  else
 10  dbms_output.put_line('largest number is:'||c);
 11  end if;
 12  else
 13  if b>c then
 14  dbms_output.put_line('largest number is:'||b);
 15  else
 16  dbms_output.put_line('largest number is:'||c);
 17  end if;
 18  end if;
 19  if a<b then
 20  if a<c then
 21  dbms_output.put_line('smallest number is:'||a);
 22  else
 23  dbms_output.put_line('smallest number is:'||c);
 24  end if;
 25  else
 26  if b>c then
 27  dbms_output.put_line('smallest number is:'||c);
 28  else
 29  dbms_output.put_line('smallest number is:'||b);
 30  end if;
 31  end if;
 32  end;
 33  /

Procedure created.

SQL> execute p2(5,4,8);
largest number is:8
smallest number is:4

PL/SQL procedure successfully completed.
```

VISWAJYOTHI COLLEGE OF ENGINEERING & TECHNOLOGY VAZHAKULAM KERALA 686 670

```
~~7~~6.1 Else
    6.1.2 If (b>c)
        6.1.1.1 Print b is smaller
    6.1.2 Else
        6.1.2.1 Print c is smaller
7. Stop.
```

RESULT

The program executed successfully and output was obtained.

[illegible]

Experiment No: 6

FACTORIAL OF A NUMBER

AIM

To write a PL/SQL program to find the factorial of a number.

ALGORITHM

```
1. Start
2. Set server output on
3. Declare variables n, fact, i
4. Read value of x
5. Set fact = 1
6. While (i<=n)
      6.1 Set fact = fact * i
      6.2 Set i = i+1
7 End while
8 Print fact
9. Stop
```

RESULT

The program was executed successfully and output was verified.

[illegible]

```
/**********************************************************************/
/*        NAME     :  SUSAN SHIBU                                    */
/*        CLASS    :  S5 CSE-A                                       */
/*        ROLL NO  :  58                                             */
/*        DATE     :  10-12-2021                                     */
/*                                                                   */
/*                    FACTORIAL OF A NUMBER                          */
/**********************************************************************/
```

PROGRAM :

```
SQL> set serveroutput on;
SQL> declare
  2  a integer;
  3  fac integer:=1;
  4  i integer;
  5  begin
  6  a:=&a;
  7  for i IN REVERSE 1..a LOOP
  8  fac:=fac*i;
  9  end loop;
 10  dbms_output.put_line(fac);
 11  end;
 12  /
```

OUTPUT :

```
Enter value for a: 5
old   6: a:=&a;
new   6: a:=5;
120

PL/SQL procedure successfully completed.
```

Experiment No:7

FIBONACCI SERIES

AIM

To write a PL/SQL program to print the fibonacci series below a given number.

ALGORITHM

```
1.  Start
2.  Set serveroutput on
3.  Declare variables first, second, third and n
    Read value of n
    Set first =0
    Set second =1
    Print first, second
    Set third = first + second
    while ( third < n)
        9.1  Print third
        9.2  Set first = second
        9.3  Set second = third
10  End of while
11  Stop
```

RESULT

✓ The program executed successfully and output was verified.

[illegible]

```
/****************************************************************************/
/*        NAME     : SUSAN SHIBU                                           */
/*        CLASS    : S5 CSE-A                                              */
/*        ROLL NO  : 58                                                    */
/*        DATE     : 10-12-2021                                            */
/*                                                                         */
/*        FIBONACCI SERIES                                                 */
/****************************************************************************/
```

PROGRAM :

```
SQL> set serveroutput on;
SQL> declare
  2  first integer:=0;
  3  second integer:=1;
  4  num integer;
  5  temp integer;
  6  begin
  7  num:=&num;
  8  if num=1
  9  then
 10  dbms_output.put_line(first);
 11  end if;
 12  if num=2
 13  then
 14  dbms_output.put_line(first);
 15  dbms_output.put_line(second);
 16  end if;
 17  if num>2
 18  then
 19  dbms_output.put_line(first);
 20  dbms_output.put_line(second);
 21  num:=num-2;
 22  while num>0 loop
 23  temp:=first+second;
 24  first:=second;
 25  second:=temp;
 26  dbms_output.put_line(second);
 27  num:=num-1;
 28  end loop;
 29  end if;
 30  end;
 31  /
```

OUTPUT :

```
Enter value for num: 5
old   7: num:=&num;
new   7: num:=5;
0
1
1
2
3

PL/SQL procedure successfully completed.
```

/ /

Experiment No: 8

REVERSE A STRING

AIM

To write a PL/SQL program to reverse a string

ALGORITHM

```
1. Start
2. Set serveroutput on
3  Declare variables word, reverse, V, l
4. Read the value of word
5  Store the length of word in l
6  While (l>0)
       6.1 Set V = substr(word, l, 1)
       6.2 Set reverse = concat(reverse, v)
       6.3 Set l = l-1
   7. End while
   8 Print reverse
   9. Stop
```

RESULT

The program executed successfully and output was verified.

```
SQL>  set serveroutput on;
/*****************************************************************************/
/*                                                                          */
/*       NAME     : SUSAN SHIBU                                              */
/*       CLASS    : S5 CSE-A                                                 */
/*       ROLL NO  : 58                                                       */
/*       DATE     : 10-12-2021                                               */
/*                                                                          */
/*******************   REVERSE STRING                                        */
/*****************************************************************************/
```

PROGRAM :

```
SQL> declare
  2  s varchar(20);
  3  r varchar(20);
  4  v varchar(2);
  5  l number:=0;
  6  begin
  7  s:='&s';
  8  l:=Length(s);
  9  while(l>0) LOOP
 10  v:=substr(s,l,1);
 11  r:=concat(r,v);
 12  l:=l-1;
 13  end LOOP;
 14  dbms_output.put_line(r);
 15  end;
 16  /
```

OUTPUT :

```
Enter value for s: susan
old   7: s:='&s';
new   7: s:='susan';
nasus

PL/SQL procedure successfully completed.
```

Experiment No: 9

INSERT ODD AND EVEN NUMBERS INTO TWO TABLES

AIM :-

To write a PL/SQL program to insert odd and even numbers from 1 to 25 into 2 tables, ODD and EVEN

ALGORITHM

```
1. Start
2. Create 2 tables ODD and EVEN
3  Declare variable number, i
4  Set i = 1
5  while (i<=25)
     5.1 If (mod (i,2) = =0)
           5.1.1 Insert i into EVEN table
     5.2 Else
           5.2.1 Insert i into ODD table
     5.3 Set i=i+1
6  Stop
```

RESULT

The program was executed successfully and output was obtained.

✓

[illegible]

```
/**************************************************************************/
/*        NAME     : SUSAN SHIBU                                           */
/*        CLASS    : S5 CSE-A                                              */
/*        ROLL NO  : 58                                                    */
/*        DATE     : 16-12-2021                                            */
/*                                                                         */
/*                              EVEN AND ODD                               */
/**************************************************************************/
```

PROGRAM :

```
CREATE TABLE ODD58(NUM NUMBER(3));
Table created.

CREATE TABLE EVEN58(NUM NUMBER(3));
Table created.


SQL> set serveroutput on;
SQL> declare
  2  a number(5):=1;
  3  r number(5):=0;
  4  begin
  5  while(a<=25)
  6  loop
  7  r:=mod(a,2);
  8  if (r=0)
  9  then
 10  insert into EVEN58 values(a);
 11  else
 12  insert into ODD58 values(a);
 13  end if;
 14  a:=a+1;
 15  end loop;
 16  end;
 17  /

PL/SQL procedure successfully completed.
****************************************************************************

SQL> SELECT * FROM ODD58;
```

OUTPUT :

```
       NUM
----------
         1
```

Experiment No: 10

PROGRAM TO UPDATE THE SALARY TABLE

AIM

To write a PL/SQL program to update the slary of Ramesh by 20% if he is earning salary greater than 8000 otherwise update by 10% if he is earning salary greater than he is earning salary greater than 5000, otherwise update by 5%

ALGORITHM

```
1. Start
2. Create table income with two columns ename
   and salary
3. Declare number S
4. Select salary into S from income where
   ename = "Ramesh"
5. If (S>=8000)
      5.1 Set S=S+(S*0.2)
6  Else if (S>=5000 and S<8000)
      6.1 Set S=S+(S*0.1)
7. Else
      7.1 Set S=S+(S*0.05)
8. Update income table by setting salary = S
   where ename = 'Ramesh'
```

```
/*********************************************************************/
/*        NAME     : SUSAN SHIBU                                     */
/*        CLASS    : S5 CSE-A                                        */
/*        ROLL NO  : 58                                              */
/*        DATE     : 09-12-2021                                      */
/*                                                                   */
/*                 INCOME PROBLEM                                    */
/*********************************************************************/
```

PROGRAM

```
CREATE TABLE INCOME58(ename varchar(20),salary number(6));

insert into INCOME58 values('Ramesh',12000);

select * from INCOME58;

declare
sal number(6);
begin
select salary into sal from INCOME58 where ename='Ramesh';
if sal>8000 then
sal:=sal+((20/100)*sal);
elsif sal>5000 then
sal:=sal+((10/100)*sal);
else
sal:=sal+((5/100)*sal);
end if;
update INCOME58 set salary =sal where ename='Ramesh';
end;
/
```

```
**********************************************************************
```

OUTPUT

```
select * from income58;

ENAME                        SALARY
-------------------------- ----------
Ramesh                        14400
```

VISWAJYOTHI COLLEGE OF ENGINEERING & TECHNOLOGY
VJCET
विद्या सर्वस्य भूषणम्
VAZHAKULAM, KERALA - 686670

9. Stop

RESULT

The program was executed successfully and output was verified.

[illegible]

/ /

Experiment No: 11

LARGEST AND SMALLEST OF THREE NUMBERS USING PROCEDURE

AIM

To write a stored procedure to find the largest and smallest of three numbers

ALGORITHM

```
1. Start
2. Create a procedure P2 with parameters
     a, b and c
3. Check if a>b, if yes
     3.1 Check if a>c if yes
           3.1.1 Display "largest number is" a
     3.2 Else
           3.2.1 Display "la gest number is" c
     3.3 End of if
4. Else
     4.1 Check if b>c if yes
           4.1.1 Display "largest number is" b
     4.2 Else
           4.2.1 Display "largest number is" c
     4.3 End of if
5 End of if
```

```
/****************************************************************************/
/*        NAME      : SUSAN SHIBU                                          */
/*        CLASS     : S5 CSE-A                                             */
/*        ROLL NO   : 58                                                   */
/*        DATE      : 20-12-2021                                           */
/*                                                                         */
/*                  LARGEST AND SMALLEST OF 3 NUMBERS                      */
/****************************************************************************/

SQL> set serveroutput on;
SQL> create or replace procedure p2(a in number,b in number,c in number)
  2  as
  3  l number;
  4  s number;
  5  begin
  6  if a>b then
  7  if a>c then
  8  dbms_output.put_line('largest number is:'||a);
  9  else
 10  dbms_output.put_line('largest number is:'||c);
 11  end if;
 12  else
 13  if b>c then
 14  dbms_output.put_line('largest number is:'||b);
 15  else
 16  dbms_output.put_line('largest number is:'||c);
 17  end if;
 18  end if;
 19  if a<b then
 20  if a<c then
 21  dbms_output.put_line('smallest number is:'||a);
 22  else
 23  dbms_output.put_line('smallest number is:'||c);
 24  end if;
 25  else
 26  if b>c then
 27  dbms_output.put_line('smallest number is:'||c);
 28  else
 29  dbms_output.put_line('smallest number is:'||b);
 30  end if;
 31  end if;
 32  end;
 33  /

Procedure created.

SQL> execute p2(5,4,8);
largest number is:8
smallest number is:4

PL/SQL procedure successfully completed.
```

/ /

```
6. Check if a<b if yes
        6.1 Check if a<c
                6.1.1 Display "smallest number is" a
        6.2 Else
                6.2.1 Display "smallest number is" c
        6.3 End of if
7. Else
        7.1 Check if b<c
                7.1.1 Display "smallest number is" b
        7.2 Else
                7.2.1 Display "smallest number is" c
        7.3 End of if
8 End of if
9. End
```

RESULT

The program was executed successfully and output was verified.

/ /

Experiment No: 12

DISPLAY GRADE FROM STUDENT DATABASE

AIM

Write a PL/SQL program to display the grade of a particular student from student database. Use a stored procedure to display the grade: >100 – A, 70-100 – B, 50-70 – C, <50 Fail

ALGORITHM

```
1. Start
2. Create a table Student580 with fields name,
   rollno: and marks
3. Insert entries into this table
4. Create a procedure P3 with parameter R
5. Define variables mark and grade
6. Store marks from the table to variable mark
   whose roll no = R
7. Check if mark >= 100, then assign grade = 'A'
8. Else check if $70 \leq$ mark $< 100$, then assign
   grade = 'B'
9. Else check if $50 \leq$ mark $< 70$, then assign
   grade = 'C'
10 Else check if mark < 50, then assign
   grade = 'F'
11 End of if
```

```
/*********************************************************************/
/*        NAME     : SUSAN SHIBU                                     */
/*        CLASS    : S5 CSE-A                                        */
/*        ROLL NO  : 58                                              */
/*        DATE     : 20-12-2021                                      */
/*                                                                   */
/*                          MARKS-GRADE                              */
/*********************************************************************/

SQL> set serveroutput on;
CREATE TABLE STUDENT580 (NAME varchar(10),ROLLNUM number(10),MARKS number(10));
INSERT INTO STUDENT580 VALUES('SUSAN',58,98);
SQL> create or replace procedure p3(R in number)
  2  as
  3  mark number(5);
  4  grade varchar(2);
  5  begin
  6  select marks into mark from student580 where rollnum=R;
  7  if mark>=100 then
  8  grade:='A';
  9  elsif mark>=70 and mark<100 then
 10  grade:='B';
 11  elsif mark>=50 and mark<700 then
 12  grade:='C';
 13  elsif mark<50 then
 14  grade:='F';
 15  end if;
 16  dbms_output.put_line('GRADE: '||grade);
 17  end;
 18  /


OUTPUT :

Procedure created.

SQL> execute p3(57);
GRADE: B

PL/SQL procedure successfully completed.
```

12 Display value of variable grade
13 End of procedure
14 Stop

RESULT

The program was executed successfully and output was verified.

/ /

Experiment No: 13

PRINT SALARY USING FUNCTION

AIM

Write a function which accepts the employeeid and prints his/her salary (employee table: employee id, salary, job)

ALGORITHM

```
1. Start
2. Create table and insert values
3. Create a function sal that returns number
4. Begin
5. Select salary into sal where empid = id
6. return (sal)
7 end
8 Accept empid from user
9 Give salary = sal (empid)
10 Print salary
11 Stop
```

RESULT

The program was successfully implemented and output was verified.

```
/*************************************************************************/
/*        NAME      :  SUSAN SHIBU                                      */
/*        CLASS     :  S5 CSE-A                                         */
/*        ROLL NO   :  58                                               */
/*        DATE      :  25-02-2022                                       */
/*                                                                      */
/*                   PRINT SALARY USING FUNCTION                        */
/*************************************************************************/

SQL> create table employeetable(id number(3), ename varchar(20), salary number(10),
job varchar(20));

SQL> create or replace function salary(empid in number) return number
  2  as
  3  sal number;
  4  begin
  5  select salary into sal from employeetable where id=empid;
  6  return sal;
  7  end;
  8  /

Function created.

SQL> declare
  2  amount number;
  3  employeeid number;
  4  begin
  5  employeeid:=&employeeid;
  6  amount:=salary(employeeid);
  7  dbms_output.put_line('The salary is '||amount);
  8  end;
  9  /

>> OUTPUT
*********************************************************************************
Enter value for employeeid: 1
old   5: employeeid:=&employeeid;
new   5: employeeid:=1;
The salary is 10000

PL/SQL procedure successfully completed.

SQL> select * from employeetable;

        ID ENAME                    SALARY JOB
---------- -------------------- ---------- --------------------
         1 john                      10000 clerk
         2 soyal                     50000 programmer
         3 susan                     75000 professor
```

/ /

Experiment No: 14

FACTORIAL OF A NUMBER USING FUNCTION

AIM

Write a program to find the factorial of a number using function.

ALGORITHM

```
1. Start
2. Create a function that returns a number
3. Begin
4      Declare and initialize f = 1, i = 1
5      Repeat while (i<=n)
           5.1  f = f * i
           5.2  i = i + 1
6.         End of loop
7    Return f
8    Invoke factorial function
9.   print factorial.
10.  Stop
```

RESULT

Program is executed and output is verified

```
/***********************************************************************/
/*        NAME     : SUSAN SHIBU                                      */
/*        CLASS    : S5 CSE-A                                         */
/*        ROLL NO  : 58                                               */
/*        DATE     : 25-02-2022                                       */
/*                                                                    */
/*                  FACTORIAL USING FUNCTION                          */
/***********************************************************************/

SQL> create or replace function factorial(fnum in number) return number
  2  as
  3  fact number:=1;
  4  fnum2 number:=0;
  5  begin
  6  fnum2:=fnum;
  7  while fnum2>0 loop
  8  fact:=fact*fnum2;
  9  fnum2:=fnum2-1;
 10  end loop;
 11  return fact;
 12  end;
 13  /

Function created.

SQL> declare
  2  num1 number;
  3  fact_result number;
  4  begin
  5  num1:=&num1;
  6  fact_result:=factorial(num1);
  7  dbms_output.put_line('The factorial is '||fact_result);
  8  end;
  9  /

>> OUTPUT
*******************************************************************************
Enter value for num1: 4
old   5: num1:=&num1;
new   5: num1:=4;
The factorial is 24

PL/SQL procedure successfully completed.
```

/ /

Experiment No: 15

DISPLAY TOTAL SALARY USING CURSOR

AIM

To write a PL/SQL program to display the salary of all employees and find the total salary paid to the employees using cursor loop

ALGORITHM

```
1. Start
2. Declare a variable total_sal = 0
3. Declare a cursor c_emp
4. Select all fields from Employee58 table to
   c_emp
5. Declare a cursor variable rw to store each
   row.
6. Begin
7.  Open c-emp and fetch each row in c-emp
    into cursor variable rw.
8.  If no more rows found in c-emp, exit loop
9.  Total_sal = total_sal + rw.salary.
10. Repeat steps 7,8,9. until no more rows left in
    c-emp.
11. Display Total_sal
12  Close the cursor c-emp
```

```
/************************************************************************/
/*                                                                      */
/*      NAME     : SUSAN SHIBU                                          */
/*      CLASS    : S5 CSE-A                                             */
/*      ROLL NO  : 58                                                   */
/*      DATE     : 10-12-2021                                           */
/*                                                                      */
/*                                       CURSOR 2                       */
/************************************************************************/
```

PRORGAM :

```
SQL> DECLARE
  2  total_sal number:=0;
  3     CURSOR c_emp is
  4        SELECT * FROM EMPLOYEE58;
  5  rw c_emp%rowtype;
  6  BEGIN
  7     OPEN c_emp;
  8     LOOP
  9     FETCH c_emp into rw;
 10        EXIT WHEN c_emp%notfound;
 11  total_sal:=total_sal+rw.salary;
 12     END LOOP;
 13  dbms_output.put_line('total salary: '||total_sal);
 14     CLOSE c_emp;
 15  END;
 16  /
total salary:57000

PL/SQL procedure successfully completed.
```

/ /

13 Stop

RESULT

The program was executed successfully and output was verified.

/ /

Experiment No: 16

PASS OR FAIL USING CURSOR

AIM

Write a PL/SQL program to create a student table(regno, name, mark1, mark2) and insert the data. Also create two tables pass and fail and insert the details of passed students into pass table and failed students into fail table

Conditions for pass :

(1) total >= 80

(2) Mark1 and Mark2 >= 35

ALGORITHM

1. Start
2. Create a table named student_15 with fields regno, name, mark1, mark2 and insert values into it
3. Create two other tables studentpass_15 and studentfail 15 with same fields.
4. Declare and initialise total_mark = 0
5. Declare a cursor c_students
6. Select all fields from student_15 table into c_students

```
/**********************************************************************/
/*        NAME     : SUSAN SHIBU                                     */
/*        CLASS    : S5 CSE-A                                        */
/*        ROLL NO  : 58                                              */
/*        DATE     : 10-12-2021                                      */
/*                                                                   */
/*                              CURSOR 1                             */
/**********************************************************************/

DECLARE
total_mark number:=0;
   CURSOR c_students is
      SELECT * FROM STUDENT_15;
rw c_students%rowtype;
BEGIN
   OPEN c_students;
   LOOP
   FETCH c_students into rw;
total_mark:=rw.mark1+rw.mark2;
   if((total_mark>=80) AND (rw.mark1>=35) AND (rw.mark2>=35)) then
INSERT INTO STUDENTPASS_15 VALUES (rw.reg_no, rw.name,rw.mark1, rw.mark2);
   else
INSERT INTO STUDENTFAIL_15 VALUES (rw.reg_no, rw.name,rw.mark1, rw.mark2);
   end if;
      EXIT WHEN c_students%notfound;
   END LOOP;
   CLOSE c_students;
END;
/

PL/SQL procedure successfully completed.

OUTPUT :

SQL> SELECT * FROM STUDENTPASS_15;

    REG_NO NAME                      MARK1      MARK2
---------- -------------------- ---------- ----------
       102 RASHIKA                      42         39
       103 VISHNU                       46         44

SQL> SELECT * FROM STUDENTFAIL_15;

    REG_NO NAME                      MARK1      MARK2
---------- -------------------- ---------- ----------
       104 DEVIKA                       40         32
       105 NIVYA                        34         30
```

```
7. Declare a cursor variable rw to store each
   row
8. Begin
9. Open c-students
10. Fetch each row in c-students into rw
11. Compute total_mark = rw.mark1 + rw.mark2
12. Check if ((total_mark >= 80) AND (rw.mark1 >= 35)
    AND (rw.mark2 >= 35))
        12.1 Insert row into studentpass_15
13 Else
        13.1 Insert row into studentfail_15
14 End if
15 Repeat steps 10, 11, 12, 13, 14 until no rows left
   in c-students
16. Close c-students
17 Stop
```

RESULT

The program was executed successfully and output was verified.

/ /

Experiment No: 17

AIM

Create a table Product(pno, pname, selling-price, cost_price). Insert values into the table. Write a PL/SQL program which raises an exception when the selling price > cost price

ALGORITHM

```
1.  Start
2.  Create a table called Product005 with fields
     pnum, pname, sellingp, costp.
3.  Insert entries into it
4.  Declare a variable nme of char type
5.  Declare an exception named exp
6   Declare a cursor pdt
7   Select all fields from Product005 into pdt.
8   Begin
9.  Declare a cursor variable i to store each row
10  Store each row to i
11. Check if i.sellingp > i.costp
        11.1  Assign nme = i.pname
        11.2  Raise exp. and exit
```

Scanned with CamScanner

```
/*************************************************************************/
/*        NAME      :  SUSAN SHIBU                                      */
/*        CLASS     :  S5 CSE-A                                         */
/*        ROLL NO   :  58                                               */
/*        DATE      :  25-02-2022                                       */
/*                                                                      */
/*                                  EXCEPTION                           */
/*************************************************************************/
```

```
SQL> CREATE TABLE PRODUCT005(PNUM NUMBER(15),PNAME VARCHAR(10),SELLINGP
NUMBER(5),COSTP NUMBER(5));
Table created.

SQL> SELECT * FROM PRODUCT005;

      PNUM PNAME        SELLINGP      COSTP
---------- ---------- ---------- ----------
         5 SOAP               35         30
         6 PEN                15         10


SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  NME VARCHAR(20);
  EXP EXCEPTION;
  CURSOR PDT IS SELECT * FROM PRODUCT005;
  BEGIN
  FOR I IN PDT
  LOOP
  IF I.SELLINGP > I.COSTP THEN
  NME:=I.PNAME;
  RAISE EXP;
  END IF;
  END LOOP;
  EXCEPTION
  WHEN EXP THEN
  DBMS_OUTPUT.PUT_LINE('SELLING PRICE>COST PRICE FOR'||NME);
  END;
  /
SELLING PRICE>COST PRICE FOR SOAP

PL/SQL procedure successfully completed.
```

/ /

```
12 End of if
13. Repeat 10, 11, 12 for each row.
14. Definion of the exception exp
        14.1 Display "selling price > cost price for" nme
15 Stop
```

RESULT

The program was executed successfully and output was obtained.

/ /

Experiment No: 18

AIM

Create a PRODUCT_MASTER table. Write a trigger that checks that quantity_on_hand (a field in the PRODUCT MASTER table) does not become negative.

ALGORITHM

```
1. Start
2. Create a table named PRODUCT_MASTER005
   with fields (pno, description, profit percent,
   quantity_on_hand, reorder, selling price, cost price)
3. Create a trigger named PRO_TRIG which is
   to be invoked before insert or update of
   quantity_on_hand field on PRODUCT_MASTER005
   for each row.
4. a is any new value of the field quantity_on_
   hand
5. if (a<0) then, raise application_error (-20001,
   'Quantity cannot be negative') and exit
6  End if
7. Insert values into table
8  Stop
```

```
/**********************************************************************/
/*      NAME     :  SUSAN SHIBU                                        */
/*      CLASS    :  S5 CSE-A                                           */
/*      ROLL NO  :  58                                                 */
/*      DATE     :  25-02-2022                                         */
/*                                                                     */
/*                                 TRIGGER                             */
/**********************************************************************/

SQL> CREATE TABLE PRODUCT_MASTER005(PNO NUMBER(15),DESCRIPTION VARCHAR(10),PP
NUMBER(5),QOH NUMBER(5
),REORDER NUMBER(5),SP NUMBER(5),CP NUMBER(5));

Table created.

SQL> create or replace trigger PRO_TRIG
  2  before insert or update of QOH on  PRODUCT_MASTER005
  3  for each row
  4  declare
  5  a number(10);
  6  begin
  7  a:=:new.QOH;
  8  if(a<0) then
  9  raise_application_error(-20001,'Qty cannot be negative');
 10  end if;
 11  end;
 12  /

Trigger created.

OUTPUT:

SQL> INSERT INTO PRODUCT_MASTER005 VALUES
(&PNO,'&DESCRIPTION',&PP,&QOH,&REORDER,&SP,&CP);
Enter value for pno: 8
Enter value for description: GOOD
Enter value for pp: 20
Enter value for qoh: -1
Enter value for reorder: 4
Enter value for sp: 300
Enter value for cp: 250
old   1: INSERT INTO PRODUCT_MASTER005 VALUES
(&PNO,'&DESCRIPTION',&PP,&QOH,&REORDER,&SP,&CP)
new   1: INSERT INTO PRODUCT_MASTER005 VALUES (8,'GOOD',20,-1,4,300,250)
INSERT INTO PRODUCT_MASTER005 VALUES (8,'GOOD',20,-1,4,300,250)
            *
ERROR at line 1:
ORA-20001: Qty cannot be negative
ORA-06512: at "STUDENT.PRO_TRIG", line 6
ORA-04088: error during execution of trigger 'STUDENT.PRO_TRIG'
```

/ /

RESULT

Program executed successfully and output was verified