



MALAD KANDIVALI EDUCATION SOCIETY'S
NAGINDAS KHANDWALA COLLEGE OF COMMERCE, ARTS &
MANAGEMENT STUDIES & SHANTABEN NAGINDAS KHANDWALA
COLLEGE OF SCIENCE
MALAD [W], MUMBAI – 64
AUTONOMOUS INSTITUTION
(Affiliated To University Of Mumbai)
Reaccredited 'A' Grade by NAAC | ISO 9001:2015 Certified

CERTIFICATE

Name: Aniket Ramesh Prajapati

Roll No: 351

Programme: BSc - CS

Semester: III

This is certified to be a bonafide record of practical works done by the above student in the college laboratory for the course **DATABASE MANAGEMENT SYSTEM – II** for the partial fulfilment of Third Semester of BSc IT/CS during the academic year 2021-22.

The journal work is the original study work that has been duly approved in the year 2021-22 by the undersigned.

External Examiner

Ms. Sweety Garg
(Subject-In-Charge)

Date of Examination: (College Stamp)

INDEX

Sr.No.	Date	Topic	Page No
1	14.7.2021	PL/SQL basics	1-6
2	4.8.2021	Control Structures	7-25
3	11.8.2021	Creating and working with Sequence	26-28
4	19.8.2021	Creating Procedures, Functions and Packages	29-36
5	1.9.2021	Creating Database Triggers	37-40
6	15.9.2021	Working with Collections	41-43
7	1.9.2021	Implementing Records	44-45
8	9.9.2021	System and User-defined Exception	46-47
9	15.9.2021	Implicit and Explicit Cursors	48-51

DBMS PRACTICAL NO: -1

1. Write a PL/SQL block to display the message “hello world”.

```
Run SQL Command Line

SQL*Plus: Release 11.2.0.2.0 Production on Thu Jul 15 16:11:11 2021

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect system
Enter password:
Connected.
SQL> set serveroutput on
SQL> begin
 2  dbms_output.put_line('hello world');
 3  end;
 4 /
hello world
```

2. Write a PL/SQL block which will read a number from the user and display it on the screen.

```
SQL> declare
 2  m int := &m;
 3  begin
 4  dbms_output.put_line(m);
 5  end;
 6 /
Enter value for m: 25
old  2: m int := &m;
new  2: m int := 25;
25

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL block to read a message from user and display it.

```
Run SQL Command Line

SQL> declare
 2  a varchar(20) := '&a';
 3  begin
 4  dbms_output.put_line(a);
 5  end;
 6 /
Enter value for a: Hey Aniket
old  2: a varchar(20) := '&a';
new  2: a varchar(20) := 'Hey Aniket';
Hey Aniket

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL block to display the area of a rectangle when length and breadth are accepted by the user.

Run SQL Command Line

```
SQL> declare
  2  len int := &len;
  3  breadth int := &breadth;
  4  area float;
  5  begin
  6  area := len * breadth;
  7  dbms_output.put_line('Area of Rectangle is ' || area);
  8  end;
  9 /
Enter value for len: 5
old   2: len int := &len;
new   2: len int := 5;
Enter value for breadth: 2
old   3: breadth int := &breadth;
new   3: breadth int := 2;
Area of Rectangle is 10

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL block to display the total number of employees.

Select Run SQL Command Line

```
SQL> set serveroutput on
SQL> declare
  2  totalemp int;
  3  begin
  4  select count(*) into totalemp from ap351_emp;
  5  dbms_output.put_line(totalemp);
  6  end;
  7 /
14

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL block to print the sum of two numbers accepted by user.

```
SQL> declare
  2   a float := &a;
  3   b float := &b;
  4   c float;
  5   begin
  6     c := a+b;
  7     dbms_output.put_line(c);
  8   end;
  9 /
Enter value for a: 10
old  2: a float := &a;
new  2: a float := 10;
Enter value for b: 20
old  3: b float := &b;
new  3: b float := 20;
30

PL/SQL procedure successfully completed.
```

7. Write a PL/SQL block to print the message ‘You can lead a horse to water but you can't make him drink’.

```
SQL> begin
  2   dbms_output.put_line('You can lead a horse to water but you can''t make him drink');
  3 end;
  4 /
You can lead a horse to water but you can't make him drink

PL/SQL procedure successfully completed.
```

8. Write a PL/SQL block to print the name and job of an employee who is working as CLERK earning salary of Rs 1100/-.

```
SQL> declare
  2   name varchar(20);
  3   job varchar(10);
  4   begin
  5     select ename,job into name, job from ap351_emp where job='CLERK' and sal=1100;
  6     dbms_output.put_line(name || ' ' || job);
  7   end;
  8 /
ADAMS CLERK

PL/SQL procedure successfully completed.
```

9. Write a PL/SQL block to calculate Simple Interest where principle, rate and time are accepted by the user.

```
SQL> declare
  2  a float;
  3  principal float := &principal;
  4  rate float := &rate;
  5  time float := &time;
  6  begin
  7  a := (principal*rate*time)/100;
  8  dbms_output.put_line('simple interest =' || a);
  9  end;
 10 /
Enter value for principal: 5000
old   3: principal float := &principal;
new   3: principal float := 5000;
Enter value for rate: 5
old   4: rate float := &rate;
new   4: rate float := 5;
Enter value for time: 2
old   5: time float := &time;
new   5: time float := 2;
simple interest =500

PL/SQL procedure successfully completed.
```

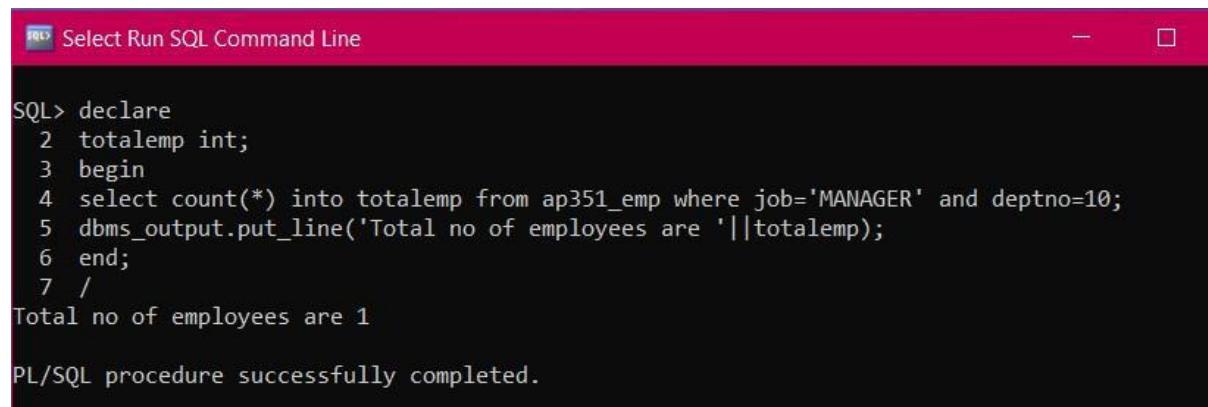
10. Write a PL/SQL block to calculate the area of the circle and store the radius and area in a table AOC (radius, area).

```
SQL> declare
  2  r int := &r;
  3  a float;
  4  begin
  5  a := 3.14*r*r;
  6  insert into AOC values(r,a);
  7  end;
  8 /
Enter value for r: 5
old   2: r int := &r;
new   2: r int := 5;
insert into AOC values(r,a);
```

11. Write a PL/SQL block to print the salary and job of the employee whose employee number is entered by the user using percent type attribute.

```
SQL> declare
 2  eno EMP.EMPNO%type :=&eno;
 3  salary EMP.SAL%type;
 4  job EMP.JOB%type;
 5  begin
 6    select sal into salary from EMP where empno = eno;
 7    select job into job from EMP where empno = eno;
 8    dbms_output.put_line('JOB:' || job);
 9    dbms_output.put_line('SALARY:' || salary);
10  end;
11 /
Enter value for eno: 5
old  2: eno EMP.EMPNO%type :=&eno;
new  2: eno EMP.EMPNO%type :=5;
```

12. Write a PL/SQL block to print the total number of employees working as Manager in deptno 10.



The screenshot shows a window titled "Select Run SQL Command Line". Inside, a PL/SQL block is run to count employees in department 10 who are managers. The output shows the count as 1, and a message at the bottom indicates the procedure was successfully completed.

```
SQL> declare
 2  totalemp int;
 3  begin
 4  select count(*) into totalemp from ap351_emp where job='MANAGER' and deptno=10;
 5  dbms_output.put_line('Total no of employees are '||totalemp);
 6  end;
 7 /
Total no of employees are 1
PL/SQL procedure successfully completed.
```

13. Write a PL/SQL block to print the total salary of the employees from the employee table

```
SQL> declare
 2  totalsal int;
 3  begin
 4  select sum(sal) into totalsal from ap351_emp;
 5  dbms_output.put_line('Total no of employees is '||totalsal);
 6  end;
 7 /
Total no of employees is 29025

PL/SQL procedure successfully completed.
```

14. Write a PL/SQL block to find the cube of a number.

```
SQL> declare
 2  cube float;
 3  a int :=5;
 4  begin
 5  cube := a*a*a;
 6  dbms_output.put_line('cube is = '||cube);
 7  end;
 8 /
cube is = 125

PL/SQL procedure successfully completed.
```

15. Write a block to print the message "I'm a user".

```
SQL> begin
 2  dbms_output.put_line('i''m a user');
 3  end;
 4 /
i'm a user

PL/SQL procedure successfully completed.

SQL>
```

PRACTICAL NO: -2

1. Write a PL/SQL program to display the sum of first 10 integers.

The screenshot shows a terminal window titled "Run SQL Command Line". The SQL*Plus session starts with the system connection command. It then declares variables a, b, and c, initializes them, and uses a for loop to iterate from 1 to 10, updating variable b with the sum of a and b in each iteration. Finally, it outputs the value of b and ends the procedure. The message "PL/SQL procedure successfully completed." is displayed at the end.

```
SQL*Plus: Release 11.2.0.2.0 Production on Mon Aug 9 17:24:11 2021
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect system
Enter password:
Connected.
SQL> declare
2   a int;
3   b int :=0;
4   begin
5     for a in 1..10 loop
6       b :=b+a;
7     end loop;
8     dbms_output.put_line(b);
9   end;
10 /
PL/SQL procedure successfully completed.
```

2. Write a PL/SQL program to display the sum of first 10 odd numbers.

The screenshot shows a terminal window titled "Run SQL Command Line". The SQL*Plus session starts with a declare block. It initializes variables a, b, and c, and then enters a while loop where a is less than 10. In each iteration, it adds the value of b to c, increments b by 2, and increments a by 1. After the loop, it outputs the value of c and ends the procedure. The output shows the odd numbers from 1 to 19, followed by the message "PL/SQL procedure successfully completed."

```
SQL> declare
2   a number :=0;
3   b number :=1;
4   c number :=0;
5   begin
6     while a<10 loop
7       dbms_output.put_line(b);
8       c :=b+c;
9       b :=b+2;
10      a :=a+1;
11    end loop;
12    dbms_output.put_line(c);
13  end;
14 /
1
3
5
7
9
11
13
15
17
19
100
PL/SQL procedure successfully completed.
```

3. Write a PL/SQL program to calculate the area of a circle and insert the area and radius in a table aoc (sno, radius, area) till radius is less than 10.

```
SQL> declare
 2  a number;
 3  radius number :=1;
 4  area number;
 5  pi number :=3.14;
 6  begin
 7  for a in 1..9 loop
 8  radius := a;
 9  area := radius * radius * pi;
10  insert into aco values(a, radius, area);
11 end loop;
12 end;
13 /
```

PL/SQL procedure successfully completed.

4. Write a PL/SQL program to reverse the number (234 as 432).

```
Run SQL Command Line
```

```
SQL> declare
 2  a number :=234;
 3  b number;
 4  begin
 5  b :=0;
 6  while a>0 loop
 7  b :=(b * 10) + mod(a,10);
 8  a :=floor(a/10);
 9  end loop;
10 dbms_output.put_line(b);
11 end;
12 /
432
```

PL/SQL procedure successfully completed.

5. Write a PL/SQL program to print the length of entered string.

```
Run SQL Command Line

SQL> declare
  2  a varchar2(20) := '&a';
  3  begin
  4  dbms_output.put_line(length(a));
  5  end;
  6 /
Enter value for a: hello
old   2: a varchar2(20) := '&a';
new   2: a varchar2(20) := 'hello';
5

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL program to count number of employees in dept 10, and if they are greater than 3 print the count otherwise do nothing.

```
Run SQL Command Line

SQL> declare
  2  num number;
  3  begin
  4  select count(*) into num from aniket_emp351 where deptno = 10;
  5  if num > 3 then
  6  dbms_output.put_line(num || 'Employee in deptno 10');
  7  end if;
  8  end;
  9 /

PL/SQL procedure successfully completed.
```

7. Write a PL/SQL block to find the factorial of a number.

```
SQL> declare
  2  a number :=&a;
  3  b number := 1;
  4  begin
  5  while a>0 loop
  6  b := b*a;
  7  a := a-1;
  8  end loop;
  9  dbms_output.put_line(b);
10 end;
11 /
Enter value for a: 4
old  2: a number :=&a;
new  2: a number :=4;
24

PL/SQL procedure successfully completed.
```

8. Write a block to display the number from 1 to 10 using unconstraint loop.

```
SQL> declare
  2  a number :=1;
  3  begin
  4  loop
  5  if a>10 then
  6  exit;
  7  end if;
  8  dbms_output.put_line(a);
  9  a :=a+1;
10 end loop;
11 end;
12 /
1
2
3
4
5
6
7
8
9
10

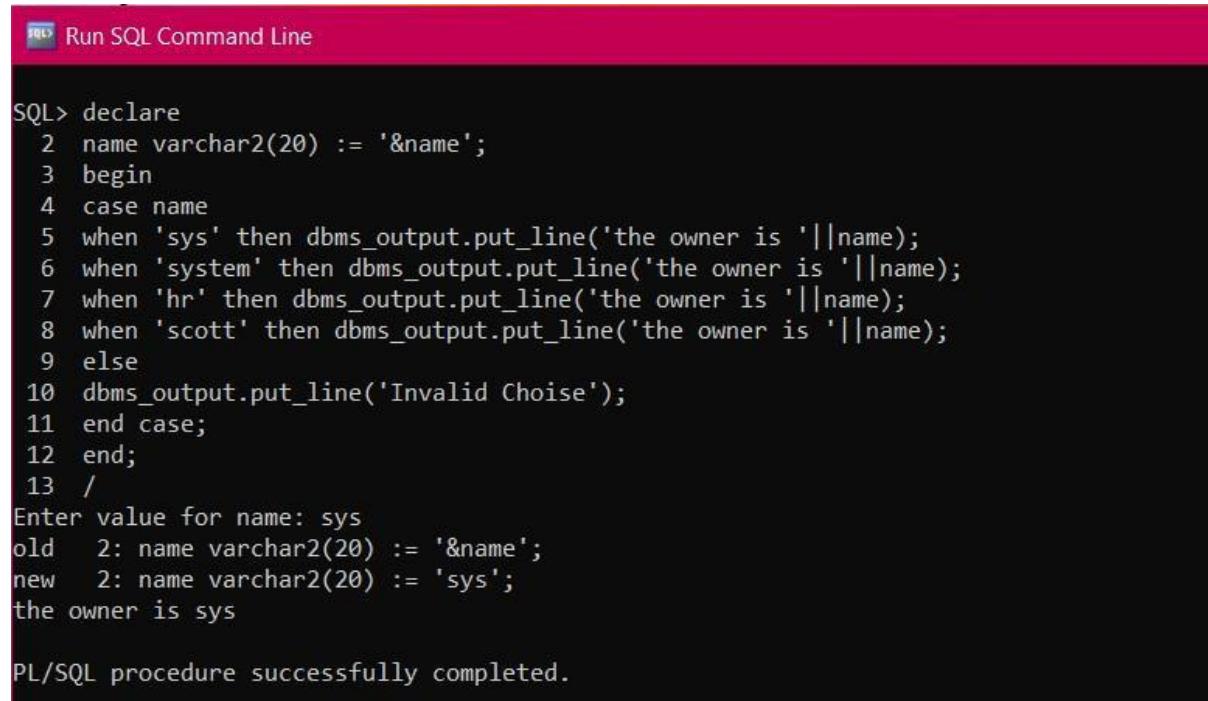
PL/SQL procedure successfully completed.
```

9. Write a PL/SQL block using CASE statement to accept the owner name from the user.

The user name can be SYS, SYSTEM, HR or SCOTT.

If the owner name is SYS then print the result is ‘The Owner is SYS’. If the owner name is SYSTEM then print the result is ‘The Owner is SYSTEM’.

If the owner name is HR then print the result is ‘The Owner is HR’. If the owner name is SCOTT then print the result is ‘The Owner is SCOTT’. Otherwise print ‘Invalid Choice’



Run SQL Command Line

```
SQL> declare
 2  name varchar2(20) := '&name';
 3  begin
 4  case name
 5  when 'sys' then dbms_output.put_line('the owner is'||name);
 6  when 'system' then dbms_output.put_line('the owner is'||name);
 7  when 'hr' then dbms_output.put_line('the owner is'||name);
 8  when 'scott' then dbms_output.put_line('the owner is'||name);
 9  else
10  dbms_output.put_line('Invalid Choise');
11  end case;
12  end;
13 /
Enter value for name: sys
old   2: name varchar2(20) := '&name';
new   2: name varchar2(20) := 'sys';
the owner is sys
PL/SQL procedure successfully completed.
```

10. Write a PL/SQL block to find factorial of a number which is accepted by the user and store it under the table fac(num,fact).

```
Run SQL Command Line

PL/SQL procedure successfully completed.

SQL> create table fac(a number,b number);
Table created.

SQL> declare
  2  a number := &a;
  3  b number := 1;
  4  c number;
  5  begin
  6    c := a;
  7    while c>0 loop
  8      b := b*c;
  9      c := c-1;
 10    end loop;
 11    insert into fac values(a,b);
 12  end;
 13 /
Enter value for a: 5
old  2: a number := &a;
new  2: a number := 5;

PL/SQL procedure successfully completed.
```

11. Write a PL/SQL to read a number and check whether it is greater than 100 or not and print appropriate message.

```
Run SQL Command Line

SQL> declare
  2  a number :=&a;
  3  begin
  4  if a>100 then
  5    dbms_output.put_line('the given number is greater than 100');
  6  else
  7    dbms_output.put_line('the given number is smaller than 100');
  8  end if;
  9  end;
 10 /
Enter value for a: 52
old  2: a number :=&a;
new  2: a number :=52;
the given number is smaller than 100

PL/SQL procedure successfully completed.
```

12. Write a PL/SQL to read the salary of an employee 7900 and display the appropriate message if it lies in the range of 1000 and 5000.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL*Plus: Release 10.2.0.3.0 - Production on Mon Aug 9 21:20:40 2021
Copyright (c) 1982, 2006, Oracle. All Rights Reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> set serveroutput on
SQL> declare
  2   s int;
  3   begin
  4     select sal into s from emp where empno=7900;
  5     if (s >=1000) and (s <=5000) then
  6       dbms_output.put_line('the salary lies in the range od 1000 tp 5000');
  7     else
  8       dbms_output.put_line('the salary does not lies in the range od 1000 tp 5000');
  9     end if;
 10   end;
 11 /
the salary does not lies in the range od 1000 tp 5000
PL/SQL procedure successfully completed.
```

13. Write a PL/SQL to swap two numbers and display the swapped numbers.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> declare
  2   a number := &a;
  3   b number := &b;
  4   temp number;
  5   begin
  6     temp := a;
  7     a := b;
  8     b := temp;
  9     dbms_output.put_line(a|| ' ' ||b);
 10   end;
 11 /
Enter value for a: 6
old  2: a number := &a;
new  2: a number := 6;
Enter value for b: 1
old  3: b number := &b;
new  3: b number := 1;
1 6
PL/SQL procedure successfully completed.
```

14. Write a PL/SQL block to update the salary of the employee with 1000 when total number of employees in a particular department is greater than 3.

 Oracle SQL*Plus
File Edit Search Options Help

```
SQL> declare
 2 deptn number;
 3 total number;
 4 begin
 5 select count(*) into total from emp where deptno = 10;
 6 if total > 3 then
 7 update emp
 8 set sal = sal +1000;
 9 select count(*) into total from emp where deptno = 20;
10 if total > 3 then
11 update emp
12 set sal = sal +1000;
13 select count(*) into total from emp where deptno = 10;
14 if total > 3 then
15 update emp
16 set sal = sal +1000;
17 end if;
18 end if;
19 end if;
20 end;
21 /
```

PL/SQL procedure successfully completed.

15. Write a PL/SQL block to delete the records of the table employee by accepting the table name from the user.

 Oracle SQL*Plus
File Edit Search Options Help

```
SQL> declare
 2 name varchar2(20) := '&name';
 3 begin
 4 delete from emp where ename =name;
 5 end;
 6 /
Enter value for name: james
old  2: name varchar2(20) := '&name';
new  2: name varchar2(20) := 'james';

PL/SQL procedure successfully completed.
```

16. Write a PL/SQL to check whether the character entered is a vowel or not.



Oracle SQL*Plus

```
File Edit Search Options Help
```

```
SQL> declare
  2  ch char :='&ch';
  3  begin
  4  if ch in ('A','E','I','O','U')then
  5  dbms_output.put_line('the character is a vowel');
  6  else
  7  dbms_output.put_line('the character is not vowel');
  8  end if;
  9  end;
10 /
Enter value for ch: h
old  2: ch char :='&ch';
new  2: ch char :='h';
the character is not vowel

PL/SQL procedure successfully completed.
```

17. Write a PL/SQL to check whether a number is even or odd.



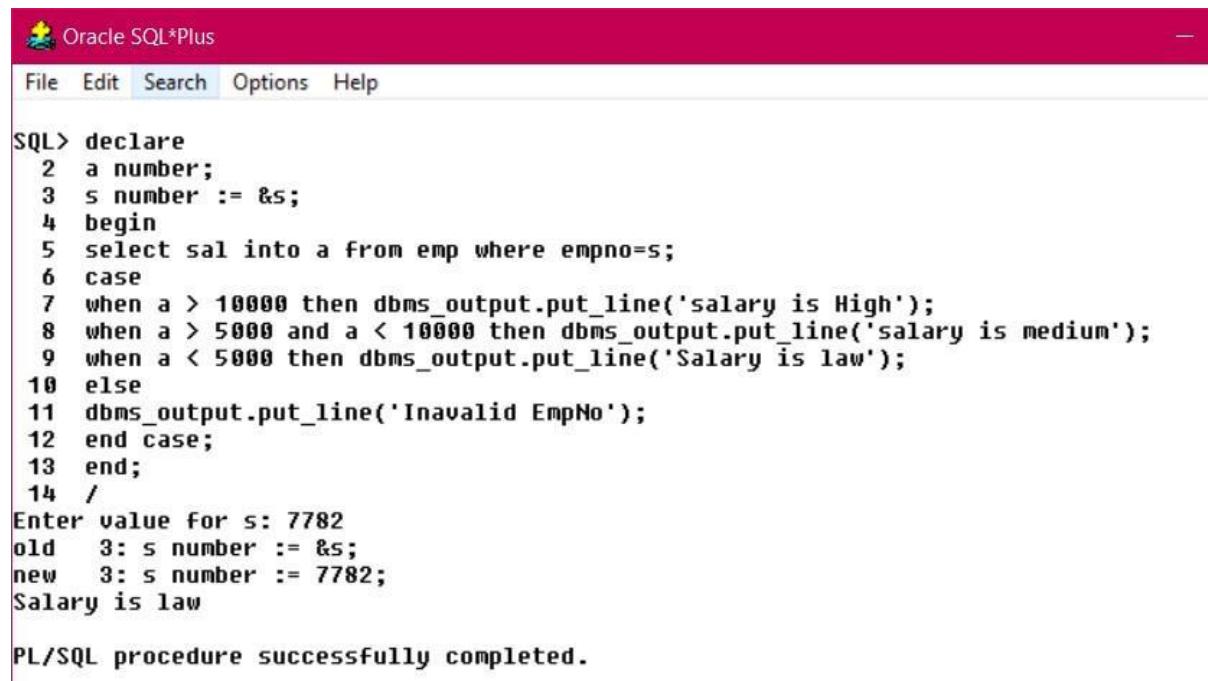
Oracle SQL*Plus

```
File Edit Search Options Help
```

```
SQL> declare
  2  a number :=&a;
  3  begin
  4  if mod(a,2) = 0 then
  5  dbms_output.put_line('the given number is even');
  6  else
  7  dbms_output.put_line('the given number is odd');
  8  end if;
  9  end;
10 /
Enter value for a: 7
old  2: a number :=&a;
new  2: a number :=7;
the given number is odd

PL/SQL procedure successfully completed.
```

18. Write a PL/SQL block using case statement to print the salary as high if it is greater than 10000, moderate if it is between 5000 and 10000 and low if it is less than 5000. The salary has been taken as an input of a specific employee whose empid is accepted by the user.

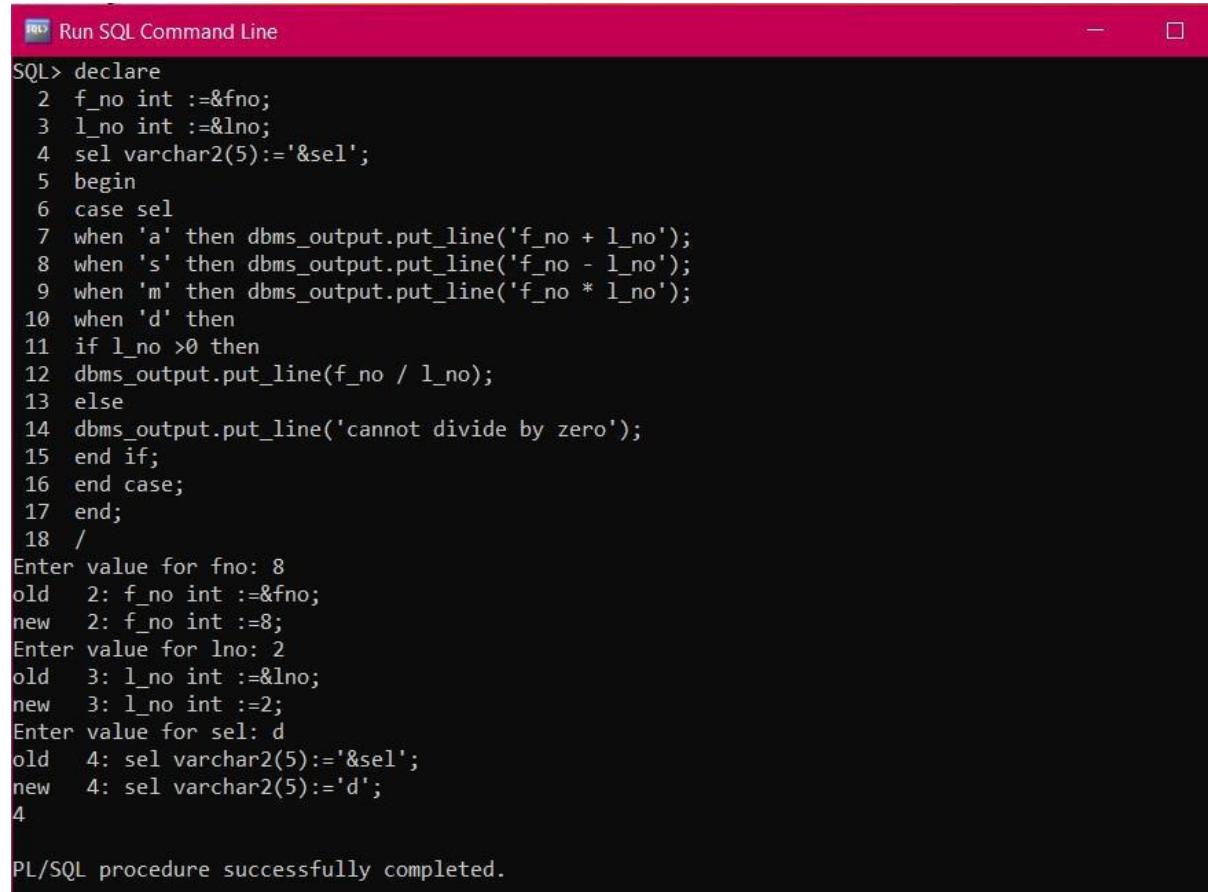


The screenshot shows the Oracle SQL*Plus interface. The menu bar includes File, Edit, Search, Options, and Help. The main area displays a PL/SQL block. The code declares a variable 'a' of type number, initializes it to the value of 's', and then performs a select operation to fetch the salary into 'a'. A case statement follows, with three when clauses: one for a > 10000 (salary is High), one for 5000 < a < 10000 (salary is medium), and one for a < 5000 (salary is low). An else clause handles invalid employee numbers. The block ends with an end case; and an end; followed by a slash. An input prompt 'Enter value for s: 7782' is shown, followed by the output of the query and the result of the case statement. The message 'PL/SQL procedure successfully completed.' is at the bottom.

```
SQL> declare
  2  a number;
  3  s number := &s;
  4  begin
  5  select sal into a from emp where empno=s;
  6  case
  7  when a > 10000 then dbms_output.put_line('salary is High');
  8  when a > 5000 and a < 10000 then dbms_output.put_line('salary is medium');
  9  when a < 5000 then dbms_output.put_line('Salary is law');
 10 else
 11 dbms_output.put_line('Inavalid EmpNo');
 12 end case;
 13 end;
 14 /
Enter value for s: 7782
old   3: s number := &s;
new   3: s number := 7782;
Salary is law

PL/SQL procedure successfully completed.
```

19. Write a PL/SQL block using case statement to perform addition, subtraction, multiplication and division for the individual choices a ,s ,m, d.The division can only take place if the divisor is greater than 0 else error message should be printed.



The screenshot shows a terminal window titled "Run SQL Command Line". The command line is executing a PL/SQL block. The block declares variables f_no, l_no, and sel, and uses a case statement to perform different operations based on the value of sel ('a', 's', 'm', or 'd'). It includes a division operation where the divisor must be greater than zero. The user is prompted to enter values for f_no, l_no, and sel. The output shows the entered values and the successful completion of the procedure.

```
SQL> declare
  2  f_no int :=&fno;
  3  l_no int :=&lno;
  4  sel varchar2(5):='&sel';
  5  begin
  6  case sel
  7  when 'a' then dbms_output.put_line('f_no + l_no');
  8  when 's' then dbms_output.put_line('f_no - l_no');
  9  when 'm' then dbms_output.put_line('f_no * l_no');
 10 when 'd' then
 11  if l_no >0 then
 12    dbms_output.put_line(f_no / l_no);
 13  else
 14    dbms_output.put_line('cannot divide by zero');
 15  end if;
 16  end case;
 17 end;
 18 /
Enter value for fno: 8
old   2: f_no int :=&fno;
new   2: f_no int :=8;
Enter value for lno: 2
old   3: l_no int :=&lno;
new   3: l_no int :=2;
Enter value for sel: d
old   4: sel varchar2(5):='&sel';
new   4: sel varchar2(5):='d';
4
PL/SQL procedure successfully completed.
```

20. Write a PL/SQL block to print the numbers from 1 to 10 using While and For Loop.

```
Run SQL Command Line

SQL> declare
  2  a number;
  3  begin
  4  for a in 1..10 loop
  5  dbms_output.put_line(a);
  6  end loop;
  7  end;
  8 /
1
2
3
4
5
6
7
8
9
10

PL/SQL procedure successfully completed.
```

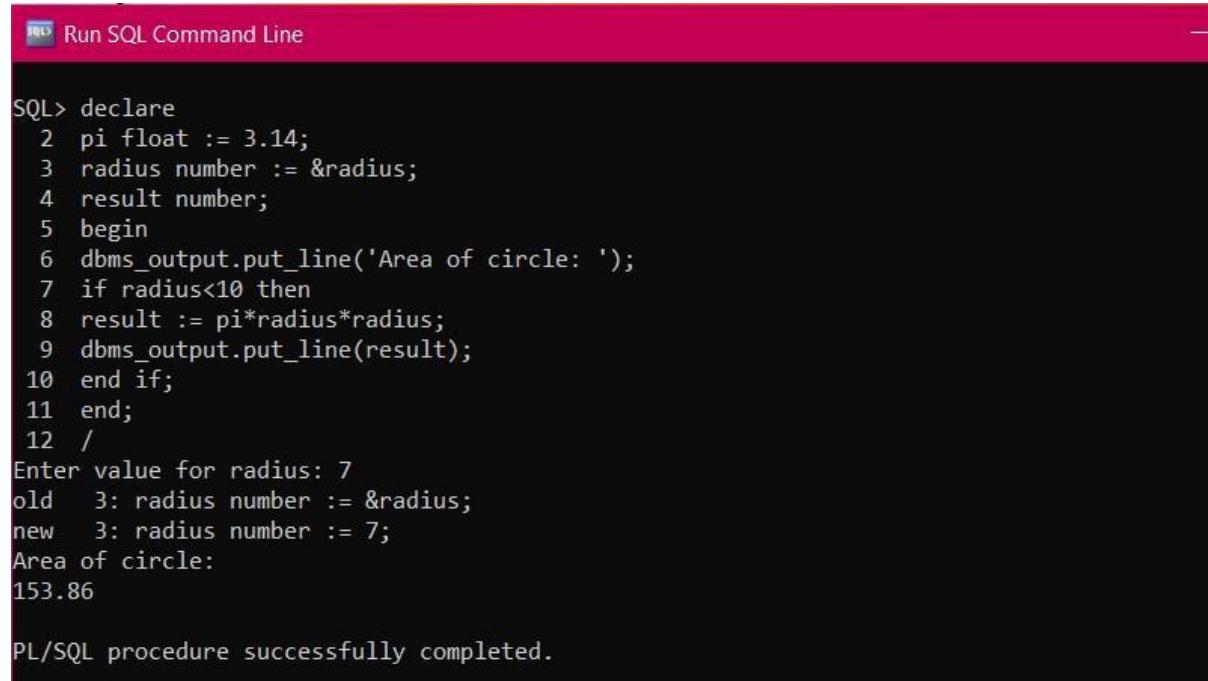
21. Write a PL/SQL block to print the Fibonacci series up to 10.

```
Run SQL Command Line

SQL> declare
  2  first number :=0;
  3  second number :=1;
  4  temp number;
  5  n number := 5;
  6  i number;
  7  begin
  8  dbms_output.put_line('Fibonacci series: ');
  9  dbms_output.put_line(first);
 10 dbms_output.put_line(second);
 11 for i in 2..n loop
 12 temp := first + second;
 13 first := second;
 14 second := temp;
 15 dbms_output.put_line(temp);
 16 end loop;
 17 end;
 18 /
Fibonacci series:
0
1
1
2
3
5

PL/SQL procedure successfully completed.
```

22. Write a PL/SQL block to calculate the area of a circle till radius less than 10. 23. Write a PL/SQL block to display the number of employees when the deptno is inputted by the user.

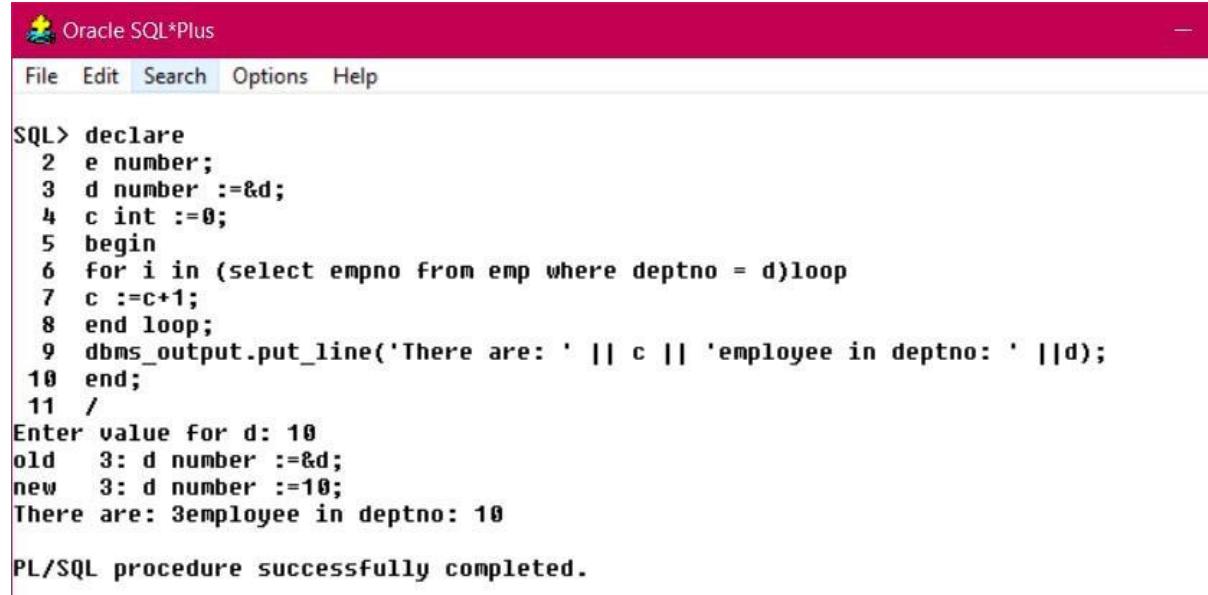


```
Run SQL Command Line

SQL> declare
  2  pi float := 3.14;
  3  radius number := &radius;
  4  result number;
  5  begin
  6  dbms_output.put_line('Area of circle: ');
  7  if radius<10 then
  8  result := pi*radius*radius;
  9  dbms_output.put_line(result);
 10 end if;
 11 end;
 12 /
Enter value for radius: 7
old   3: radius number := &radius;
new   3: radius number := 7;
Area of circle:
153.86

PL/SQL procedure successfully completed.
```

23. Write a PL/SQL block to display the number of employees when the deptno is inputted by the user.



```
Oracle SQL*Plus

File Edit Search Options Help

SQL> declare
  2  e number;
  3  d number :=&d;
  4  c int :=0;
  5  begin
  6  for i in (select empno from emp where deptno = d)loop
  7  c :=c+1;
  8  end loop;
  9  dbms_output.put_line('There are: ' || c || 'employee in deptno: ' ||d);
 10 end;
 11 /
Enter value for d: 10
old   3: d number :=&d;
new   3: d number :=10;
There are: 3employee in deptno: 10

PL/SQL procedure successfully completed.
```

24. Write a PL/SQL block to print greatest among three numbers.

```
Select Run SQL Command Line
SQL> declare
  2  a number := &a;
  3  b number := &b;
  4  c number := &c;
  5  begin
  6  if(a>b) and (a>c) then
  7    dbms_output.put_line('Number ' || a || ' is greater');
  8  elsif (b>a) and (b>c) then
  9    dbms_output.put_line('Number ' || b || ' is greater');
 10 else
 11   dbms_output.put_line('Number ' || c || ' is greater');
 12 end if;
 13 end;
 14 /
Enter value for a: 16
old  2: a number := &a;
new  2: a number := 16;
Enter value for b: 7
old  3: b number := &b;
new  3: b number := 7;
Enter value for c: 34
old  4: c number := &c;
new  4: c number := 34;
Number 34 is greater
PL/SQL procedure successfully completed.
```

25. Write a PL/SQL block to display the appropriate day of the week according to the choice made by the user.

```
Select Run SQL Command Line
SQL> declare
  2  dnum number :=&dnum;
  3  begin
  4  case dnum
  5  when 1 then dbms_output.put_line('sunday');
  6  when 2 then dbms_output.put_line('Monday');
  7  when 3 then dbms_output.put_line('Tuesday');
  8  when 4 then dbms_output.put_line('Wednesday');
  9  when 5 then dbms_output.put_line('Thursday');
 10 when 6 then dbms_output.put_line('Friday');
 11 when 7 then dbms_output.put_line('Saturday');
 12 else dbms_output.put_line('Not a number of day in week');
 13 end case;
 14 end;
 15 /
Enter value for dnum: 7
old  2: dnum number :=&dnum;
new  2: dnum number :=7;
Saturday
PL/SQL procedure successfully completed.
```

26. Create a PL/SQL block that has four sections. Each section should output a statement. Use labels and the Goto command to output the section messages in the following order:

Section 3

Section 2

Section 1

Section 4

```
Select Run SQL Command Line

SQL> Begin
 2  Goto section3;
 3  <<section1>>
 4  dbms_output.put_line('section 1');
 5  Goto section4;
 6  <<section2>>
 7  dbms_output.put_line('section 2');
 8  Goto section1;
 9  <<section3>>
10  dbms_output.put_line('section 3');
11  Goto section2;
12  <<section4>>
13  dbms_output.put_line('section 4');
14  end;
15 /
section 3
section 2
section 1
section 4

PL/SQL procedure successfully completed.
```

27. Write a PL/SQL block to check whether the entered year is a leap year or not.

```
Select Run SQL Command Line

SQL> declare
 2  year number := &year;
 3  begin
 4  if mod(year,4)=0 and mod(year,100)!=0 and mod(year,400)=0 then
 5  dbms_output.put_line(year || ' is a leap year');
 6  else
 7  dbms_output.put_line(year || ' is not a leap year');
 8  end if;
 9  end;
10 /
Enter value for year: 2015
old  2: year number := &year;
new  2: year number := 2015;
2015is not a leap year

PL/SQL procedure successfully completed.
```

28. Write a PL/SQL block to display the numbers from 1 to 10 using EXIT and EXIT WHEN statement.

```
SQL> declare
  2  a int :=0;
  3  begin
  4  loop
  5  dbms_output.put_line(a);
  6  if a>=10 then
  7  exit;
  8  end if;
  9  a:= a+1;
10  end loop;
11  end;
12 /  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
  
PL/SQL procedure successfully completed.
```

```
SQL> declare
  2  a int :=0;
  3  begin
  4  loop
  5  dbms_output.put_line(a);
  6  exit when a>10;
  7  a := a+1;
  8  end loop;
  9  end;
10 /  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
  
PL/SQL procedure successfully completed.
```

29. Write a PL/SQL block to accept job from EMP table. Give the following raise in the salary: -

By 9% if job is clerk. By 8% if job is manager.

By 7% if job is salesman.

Update the salary of the EMP table.

```
SQL> declare
  2  a varchar2(10):='&a';
  3  begin
  4  case a
  5  when 'CLERK' then
  6  update emp
  7  set sal+=(sal*(9/100)) where job=a;
  8  dbms_output.put_line('salary is updated');
  9  when 'MANAGER' then
 10  update emp
 11  set sal=sal+(sal*(8/100)) where job=a;
 12  dbms_output.put_line('salary is updated');
 13  when 'SALESMAN' then
 14  update emp
 15  set sal=sal+(sal*(7/100)) where job=a;
 16  dbms_output.put_line('salary is updated');
 17  end case;
 18  end;
 19 /
Enter value for a: MANAGER
old   2: a varchar2(10):='&a';
new   2: a varchar2(10):='MANAGER';

PL/SQL procedure successfully completed.
```

```
SQL>
```

30. Write a PL/SQL block to get the details of marks (rollno, marks1, marks2, grade) out of 100 for marks1 and marks2 respectively.

Display the grade in table marks using if statement as specified below If stud_percent > 70 then grade is ‘A’

If stud_percent > 60 and <70 then grade is ‘B’ else give grade ‘C’.

```
SQL> declare
  2   roll number := &roll;
  3   m1 number := &m1;
  4   m2 number := &m2;
  5   grade char;
  6   stud_percent number;
  7   begin
  8   stud_percent := ((m1+m2)/200) * 100;
  9   if stud_percent > 70 then
10   insert into mark values(roll,m1,m2,'A');
11  elsif stud_percent >60 and stud_percent > 70 then
12  insert into mark values(roll,m1,m2,'B');
13  else
14  insert into mark values(roll,m1,m2,'C');
15  end if;
16  end;
17 /
```

Enter value for roll: 7
old 2: roll number := &roll;
new 2: roll number := 7;
Enter value for m1: 80
old 3: m1 number := &m1;
new 3: m1 number := 80;
Enter value for m2: 90
old 4: m2 number := &m2;
new 4: m2 number := 90;

PL/SQL procedure successfully completed.

31. Write a PL/SQL block to book a ticket for a movie. The tickets are of two type's deluxe rows (D) and Ordinary rows (O).

While booking the ticket the customer may ask 'D' or 'O' and number of ticket. For deluxe the rate is 350 and for ordinary 200.

Find the total amount that the customer will pay and number of tickets (using case statement).

```
SQL> declare
  2  t_type char(1) := '&t_type';
  3  no_tickets number := '&no_tickets';
  4  begin
  5  case t_type
  6  when 'd' then dbms_output.put_line('Total price: ' || no_tickets*350);
  7  when 'o' then dbms_output.put_line('Total price: ' || no_tickets*200);
  8  else
  9  dbms_output.put_line('Not a valid ticket type');
 10 end case;
 11 end;
 12 /
Enter value for t_type: o
old   2: t_type char(1) := '&t_type';
new   2: t_type char(1) := 'o';
Enter value for no_tickets: 2
old   3: no_tickets number := '&no_tickets';
new   3: no_tickets number := '2';

PL/SQL procedure successfully completed.
```

PRACTICAL NO: -3

1. Write a PL/SQL block to create a sequence by using cycle and insert the values in a table, altering sequences.

The screenshot shows a terminal window titled "Oracle SQL*Plus". The menu bar includes "File", "Edit", "Search", "Options", and "Help". The main area displays the following SQL commands and their results:

```
SQL> create table seq(sno number,seq_num varchar2(10));
Table created.

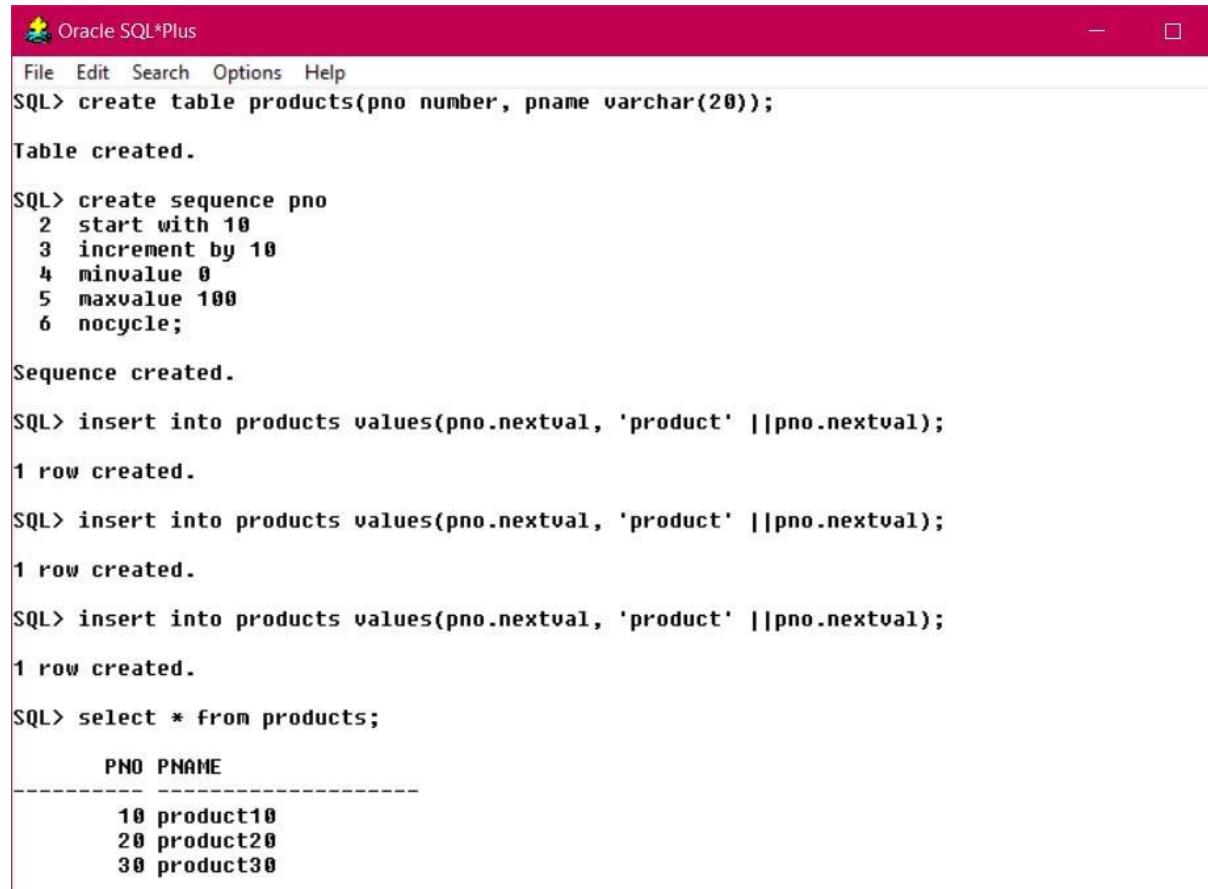
SQL> create sequence p1
  2  start with 1
  3  increment by 1
  4  minvalue 0
  5  maxvalue 50
  6  cycle;
Sequence created.

SQL> insert into seq values(p1.nextval,'seq 1');
1 row created.

SQL> insert into seq values(p1.nextval,'seq 2');
1 row created.

SQL> select * from seq;
  SNO SEQ_NUM
----- -----
    1 seq 1
    2 seq 2
```

2. Write a sequence as 10, 20, 30 100 and bind it with the table product (product no, product name).



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create table products(pno number, pname varchar(20));
Table created.

SQL> create sequence pno
  2  start with 10
  3  increment by 10
  4  minvalue 0
  5  maxvalue 100
  6  nocycle;

Sequence created.

SQL> insert into products values(pno.nextval, 'product' || pno.nextval);
1 row created.

SQL> insert into products values(pno.nextval, 'product' || pno.nextval);
1 row created.

SQL> insert into products values(pno.nextval, 'product' || pno.nextval);
1 row created.

SQL> select * from products;
  PNO  PNAME
----- 
    10 product10
    20 product20
    30 product30
```

3. Write a sequence who's maximum value is 40 and is incremented by 4, starts with 1 and forming a cycle.

```
SQL> create sequence proll
  2  start with 1
  3  increment by 4
  4  maxvalue 40
  5  cycle
  6  cache 5;

Sequence created.

SQL> create table prolls(no number);

Table created.

SQL> insert into prolls values(proll.nextval);

1 row created.

SQL> select * from prolls;

    NO
-----
     1

SQL> insert into prolls values(proll.nextval);

1 row created.

SQL> insert into prolls values(proll.nextval);

1 row created.

SQL> select * from prolls;

    NO
-----
     1
     5
     9

SQL> |
```

PRACTICAL NO: -4

1.Create and replace an empty procedure and call it.

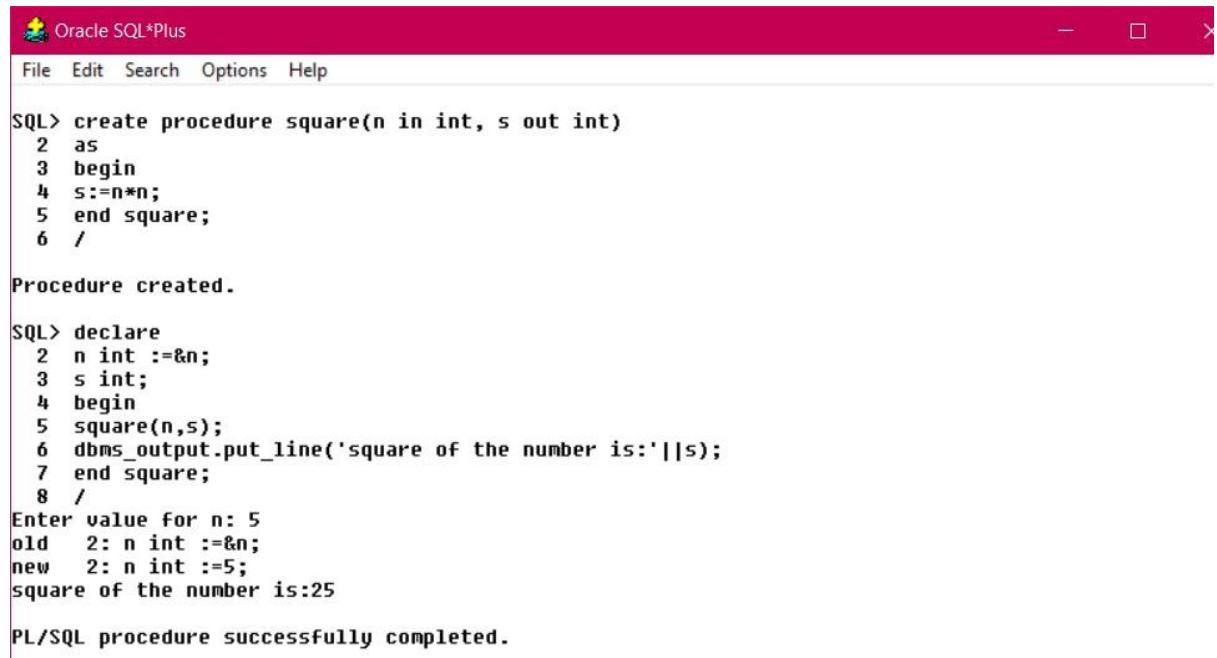
```
SQL> set serveroutput on
SQL> create procedure aniket as
 2 begin
 3 null;
 4 end;
 5 /
Procedure created.

SQL> execute aniket;
PL/SQL procedure successfully completed.

SQL> create or replace procedure aniket as
 2 begin
 3 dbms_output.put_line('Hello Aniket');
 4 end;
 5 /
Procedure created.

SQL> execute aniket;
Hello Aniket
```

2. Create procedure and function to display square of number.



The screenshot shows the Oracle SQL*Plus interface. The menu bar includes File, Edit, Search, Options, and Help. The main window displays the following PL/SQL code:

```
SQL> create procedure square(n in int, s out int)
 2 as
 3 begin
 4   s:=n*n;
 5 end square;
 6 /
Procedure created.

SQL> declare
 2   n int :=&n;
 3   s int;
 4 begin
 5   square(n,s);
 6   dbms_output.put_line('square of the number is:'||s);
 7 end square;
 8 /
Enter value for n: 5
old  2: n int :=&n;
new  2: n int :=5;
square of the number is:25

PL/SQL procedure successfully completed.
```

3. Create a procedure and a function to swap two numbers.

```
SQL> create procedure swap(a in out int, b in out int)is
  2   c number;
  3 begin
  4   c :=a;
  5   a :=b;
  6   b :=c;
  7 end;
  8 /
Procedure created.

SQL> declare
  2   a int :=4;
  3   b int :=15;
  4   c int;
  5 begin
  6   swap(a,b);
  7   dbms_output.put_line(a||' '||b);
  8 end;
 9 /
15 4

PL/SQL procedure successfully completed.
```

4. Create a procedure and a function to display greatest among two.

```
SQL> create procedure greater_num(a in out number,b in out number)
  2 as
  3 begin
  4 if a>b then
  5   dbms_output.put_line('greater number is:' ||a);
  6 else
  7   dbms_output.put_line('greater number is:' ||b);
  8 end if;
  9 end;
 10 /
Procedure created.

SQL> declare
  2   a int:=&a;
  3   b int:=&b;
  4 begin
  5   greater_num(a,b);
  6 end;
 7 /
Enter value for a: 15
old  2: a int:=&a;
new  2: a int:=15;
Enter value for b: 47
old  3: b int:=&b;
new  3: b int:=47;
greater number is:47

PL/SQL procedure successfully completed.
```



Oracle SQL*Plus

File Edit Search Options Help

```
SQL> create or replace function greater(a in int, b in int)
  2  return int
  3  as
  4  begin
  5  if a>b then
  6  return a;
  7  else
  8  return b;
  9  end if;
10  end;
11 /
```

Function created.

```
SQL> select greater(50,47) from dual;
```

```
GREATERTOOL
```

```
-----
```

```
50
```

5. Create a procedure and a function to display the employee name whose employee no is accepted by user.



Oracle SQL*Plus

File Edit Search Options Help

```
SQL> create procedure display_name(eno in int)
  2  as
  3  name varchar(20);
  4  begin
  5  select ename into name from emp where empno=eno;
  6  dbms_output.put_line(name);
  7  end;
  8 /
```

Procedure created.

```
SQL> execute display_name(7839);
```

```
KING
```

PL/SQL procedure successfully completed.

6. Create procedure and a function to display the sum of salary of the employees whose job is accepted by the user.



Oracle SQL*Plus

File Edit Search Options Help

```
SQL> create procedure salary(j in varchar)
  2  as
  3  sal int;
  4  begin
  5  select sum(sal) into sal from emp where j=job;
  6  dbms_output.put_line(sal);
  7  end;
  8 /
```

Procedure created.

```
SQL> execute salary('MANAGER');
```

```
8275
```

PL/SQL procedure successfully completed.

7. Create a procedure to display today's date.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create procedure today_date
  2  as
  3  m date;
  4  begin
  5  select sysdate into m from dual;
  6  dbms_output.put_line(m);
  7  end;
  8 /
Procedure created.

SQL> execute today_date;
13-AUG-21

PL/SQL procedure successfully completed.
```

8. Create a procedure to find the factorial of a number.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create procedure factorial(num in int)
  2  as
  3  Fact int :=1;
  4  begin
  5  for i in 1..num loop
  6  Fact := Fact * i;
  7  end loop;
  8  dbms_output.put_line(Fact);
  9  end;
 10 /
Procedure created.

SQL> execute factorial(6);
720

PL/SQL procedure successfully completed.
```

9. Create a procedure to display length of a number.

```
SQL> create procedure stringlength(str in varchar)
  2  as
  3  m date;
  4  begin
  5  dbms_output.put_line(length(str));
  6  end;
  7 /
Procedure created.

SQL> execute stringlength('HEY ANIKET');
10

PL/SQL procedure successfully completed.
```

10. Create a function to print the reverse of a string.

 Oracle SQL*Plus
File Edit Search Options Help
SQL> create function reverse
2 return number
3 as
4 rev number :=0;
5 n number:=&n;
6 begin
7 while (n>0) loop
8 rev := rev*10+mod(n,10);
9 n :=floor(n/10);
10 end loop;
11 return rev;
12 end;
13 /
Enter value for n: 598
old 5: n number:=&n;
new 5: n number:=598;

Function created.

SQL> select reverse from dual;

REVERSE

895

11. Create a package with function and procedure to fine the sum of 10 natural numbers.

 Oracle SQL*Plus
File Edit Search Options Help
SQL> create or replace package pack_practical
2 as
3 procedure sum(a in int);
4 function sumno(a int)return int;
5 end pack_practical;
6 /

Package created.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create or replace package body pack_practical
  2  as
  3  procedure sum(a in int)
  4  as
  5  begin
  6  dbms_output.put_line((a*(a+1))/2);
  7  end;
  8
  9  function sumno(a int) return int
 10 as
 11 begin
 12 return(a*(a+1))/2;
 13 end;
 14 end pack_practical;
 15 /
Package body created.
```

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> execute pack_practical.sum(10);
55
PL/SQL procedure successfully completed.
SQL> select pack_practical.sumno(10) from dual;
PACK_PRactical.SUMNO(10)
-----
      55
```

12. Create a package with a function and procedure to print the prime numbers between 1 to 50.

The screenshot shows the Oracle SQL*Plus interface with the following session history:

```
File Edit Search Options Help
SQL> create or replace package pack_prime
  2  as
  3  procedure proc(a in int);
  4  function func(a int) return varchar;
  5  end pack_prime;
  6  /
Package created.

SQL> create or replace package body pack_prime
  2  as
  3
  4  procedure proc(a in int)
  5  as
  6  b number :=0;
  7  c number :=2;
  8  begin
  9  while c<a loop
10  if remainder(a,c)=0 then
11  b:=1;
12  exit;
13  end if;
14  c:=c+1;
15  end loop;
16  if b=0 then
17  dbms_output.put_line(a);
18  end if;
19  end;
20
21  function func(a int)return varchar
22  as
23  b number:=0;
24  c number:=2;
25  prime varchar(130):='';
26  begin
27  while c<a loop
28  if remainder(a,c)=0 then
29  b:=1;
30  exit;
31  end if;
32  c := c+1;
33  end loop;
34  if b=0 then
35  prime :=prime||a;
36  end if;
37  return prime;
38  end;
39  end pack_prime;
40  /
Package body created.
```

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> begin
  2  for i in 2..50 loop
  3  pack_prime.proc(i);
  4  end loop;
  5  end;
  6 /
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
PL/SQL procedure successfully completed.

SQL> declare
  2  prime varchar(130);
  3  begin
  4  for i in 2..50 loop
  5  prime := pack_prime.func(i);
  6  dbms_output.put_line(prime);
  7  end loop;
  8  end;
  9 /
2
3
5
7
<   > ...:11
13
17
19
23
29
31
37
41
43
47
PL/SQL procedure successfully completed.

SQL>
```

PRACTICAL NO: -5

Create the following schemas:

Employee (eno, name, hrs, salary, project_no)

Project (pno, project name, thrs)

where thrs is the total hours and is the derived attribute. Its value is the sum of hrs of all employees working on that project. Also, in the given schemas eno and pno are primary keys. (for eg, if E1 is working on Project_no 10 for 12 hr and E2 is also working on Project_no 10 for 20 hr, than thrs will be updated automatically to 32 hrs)

1. To update the total hours of project when hours of employee has been updated for that particular project.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> create table projects(pno int primary key,projects_name varchar(25),thrs int);
Table created.

SQL> create table employeee(eno int,e_name varchar(20), hrs int, salary int, projects_no int references projects(pno));
Table created.

SQL> create or replace trigger updt_thrs
  2  after update on employeee
  3  referencing old as o new as n
  4  for each row
  5  begin
  6  UPDATE projects SET thrs = thrs + :n.hrs WHERE pno= :o.projects_no;
  7  end;
  8 /

Trigger created.

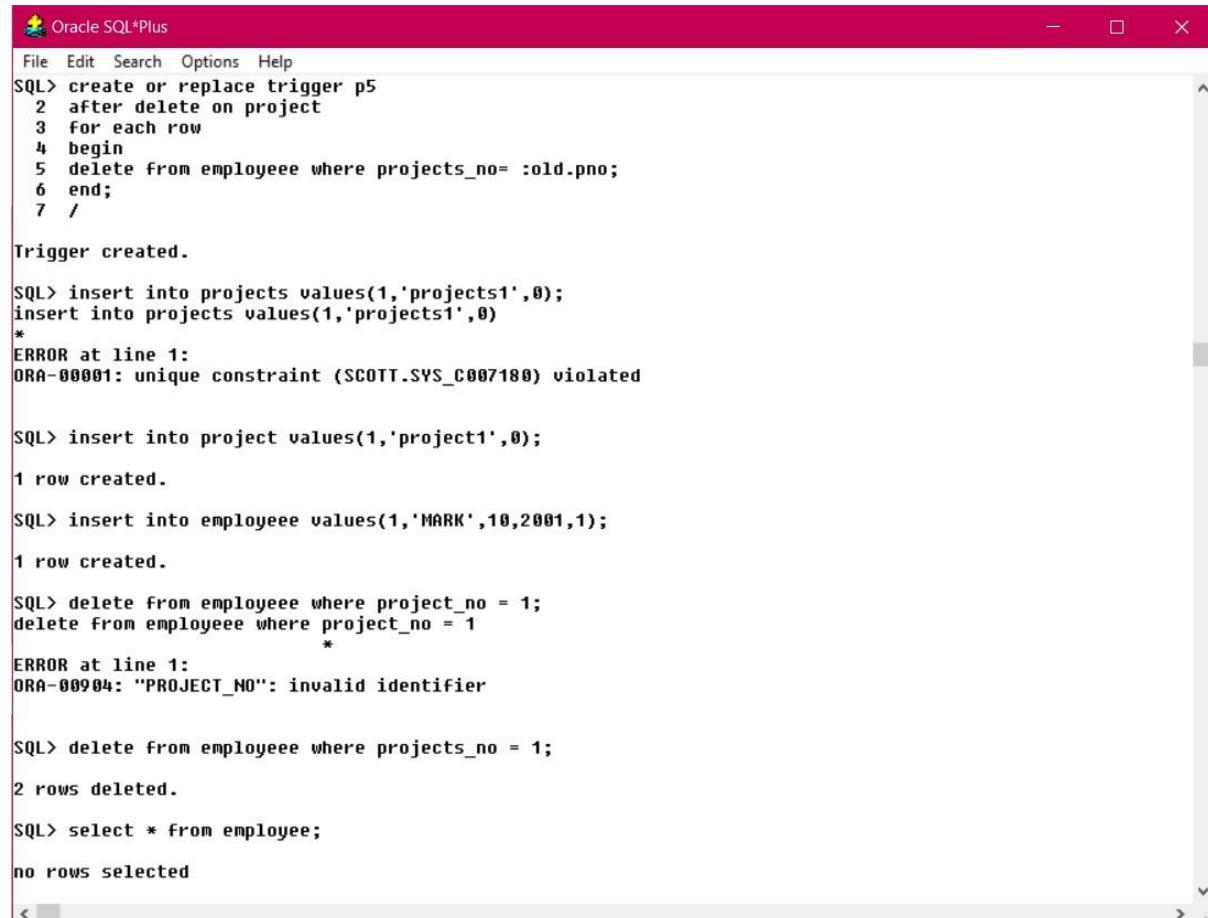
SQL> insert into projects values(1,'projects 1',0);
1 row created.

SQL> insert into employeee values(1,'Aniket',10,3000,1);
1 row created.

SQL> update employeee set hrs=hrs+10 where projects_no=1;
1 row updated.

SQL> select * from projects;
      PNO PROJECTS_NAME          THRS
----- 1 projects 1              20
```

2. Creating a trigger to delete the employees wherever corresponding project is deleted from project table.



The screenshot shows a session in Oracle SQL*Plus. The user creates a trigger named p5 that fires after a delete operation on the project table, specifically targeting the employee table where the project number matches the deleted row. After creating the trigger, the user attempts to insert two rows into the projects table, which fails due to a unique constraint violation (ORA-00001). Then, they insert a row into the project table and a row into the employee table. Finally, they attempt to delete a row from the employee table where project_no = 1, but receive an error because project_no is an invalid identifier.

```
File Edit Search Options Help
SQL> create or replace trigger p5
2  after delete on project
3  for each row
4  begin
5  delete from employeee where projects_no= :old.pno;
6  end;
7 /
Trigger created.

SQL> insert into projects values(1,'projects1',0);
insert into projects values(1,'projects1',0)
*
ERROR at line 1:
ORA-00001: unique constraint (SCOTT.SYS_C007180) violated

SQL> insert into project values(1,'project1',0);
1 row created.

SQL> insert into employeee values(1,'MARK',10,2001,1);
1 row created.

SQL> delete from employeee where project_no = 1;
delete from employeee where project_no = 1
*
ERROR at line 1:
ORA-00904: "PROJECT_NO": invalid identifier

SQL> delete from employeee where projects_no = 1;
2 rows deleted.

SQL> select * from employee;
no rows selected
```

3. Create a trigger to insert the name of the employee in UPPERCASE in a table.

The screenshot shows a session in Oracle SQL*Plus. The user creates a trigger named p5_1 that converts the employee name to uppercase before insertion. The trigger is created successfully. When attempting to insert a new row into the employee table, it fails because the primary key 'ENo' already exists. Finally, the user inserts a row into the employee table with the name 'john', and the query is successful.

```
File Edit Search Options Help
SQL> create or replace trigger p5_1
  2  before insert on employeee
  3  for each row
  4  begin
  5  :new.e_name := upper(:new.e_name);
  6  end;
  7 /

Trigger created.

SQL> insert into project values(2,'project1',0);

1 row created.

SQL> insert into employeee values(1,'john',10,3000,2);
insert into employeee values(1,'john',10,3000,2)
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.SYS_C007181) violated - parent key not
found

SQL> insert into employee values(1,'john',10,3000,2);

1 row created.

SQL> select * from employee;

ENO E_NAME          HRS    SALARY PROJECT_NO
-----  -----
 1 john              10      3000          2
```

4. Create a trigger to update the salary of employees and to show the old salary, new salary and the difference.

```

SQL> create or replace trigger p5_q4
  2  after update on employee
  3  for each row
  4  declare
  5  new_sal number;
  6  begin
  7  new_sal := :old.salary - :new.salary;
  8  dbms_output.put_line('Old salary : ' || :old.salary);
  9  dbms_output.put_line('New salary : ' || :new.salary);
 10 dbms_output.put_line('Difference in salary : ' || new_sal);
 11 end;
 12 /
Trigger created.

SQL> update employee set salary = salary - 2000 where eno = 1;
Old salary : 3000
New salary : 1000
Difference in salary : 2000
1 row updated.

```

5. Create a trigger to delete the record of an employee and insert the deleted record in DEL_EMP_REC table. In this case, DEL_EMP_REC is table which will hold the deleted records of an employee.

```

SQL> create or replace trigger p5_q5
  2  after delete on employee
  3  for each row
  4  begin
  5  insert into DEL_EMP_REC values(:old.eno,:old.salary,:old.hrs,:old.e_name,:old.project_no);
  6  end;
  7  /
Trigger created.

SQL> delete from employee where eno = 1;
1 row deleted.

SQL> select * from DEL_EMP_REC;
      ENO       SAL      HRS E_NAME      PROJECT_NO
-----  -----  -----  -----  -----
        1     5000      10  SMITH            3
SQL>

```

PRACTICAL NO: -6

1. Create a nested table with 5 integers and print them using collection method first and last. Also print total integers in a table.

```
SQL> declare
  2   type numerical is table of int;
  3   n numerical;
  4   begin
  5   n := numerical(1,2,3,4,5);
  6   dbms_output.put_line('Total Integers in list::'||n.count);
  7   for i in n.first..n.last loop
  8   dbms_output.put_line('Numbers: '||n(i));
  9   end loop;
 10  end;
 11 /
Total Integers in list::5
Numbers: 1
Numbers: 2
Numbers: 3
Numbers: 4
Numbers: 5

PL/SQL procedure successfully completed.
```

2. Using Varray, create a table of 5 elements and print them.



The screenshot shows the Oracle SQL*Plus interface with a red header bar. The menu bar includes File, Edit, Search, Options, Help, and a logo. Below the menu is a command window containing the following PL/SQL code:

```
SQL> declare
  2   type cars is varray(5) of varchar(20);
  3   c cars;
  4   begin
  5   c:=cars('Audi','BMW','JAGUAR','Rolls Royle');
  6   for i in c.first..c.last loop
  7   dbms_output.put_line('Cars: '||c(i));
  8   end loop;
  9   end;
 10 /
Cars: Audi
Cars: BMW
Cars: JAGUAR
Cars: Rolls Royle

PL/SQL procedure successfully completed.
```

3. Create a table of 10 colors, print them. Delete the color at location 5 and print the remaining elements.

```

Oracle SQL*Plus
File Edit Search Options Help
SQL> declare
  2 type color is table of varchar(20);
  3 c color;
  4 begin
  5   c:=color('red','green','blue','black','white','pink','brown','violet','biege','yellow');
  6   for i in c.first..c.last loop
  7     dbms_output.put_line('Color: '||c(i));
  8   end loop;
  9   dbms_output.put_line('-----');
10  c.delete(5);
11  for i in 1..4 loop
12    dbms_output.put_line('color: '||c(i));
13  end loop;
14  for i in 6..10 loop
15    dbms_output.put_line('color: '||c(i));
16  end loop;
17 end;
18 /
Color: red
Color: green
Color: blue
Color: black
Color: white
Color: pink
Color: brown
Color: violet
Color: biege
Color: yellow
-----
color: red
color: green
color: blue
color: black
color: pink
color: brown
color: violet
color: biege
color: yellow
PL/SQL procedure successfully completed.
SQL> |

```

4. Using Type table, print the name of the employees.

```

SQL> declare
  2 cursor c4 is select ename from emp;
  3 type nam is table of emp.ename%type index by binary_integer;
  4 names nam;
  5 c integer := 0;
  6 begin
  7   for i in c4 loop
  8     c := C+1;
  9     names(c) := i.ename;
10   dbms_output.put_line('Ename'||c||'):'||names(c));
11 end loop;
12 end;
13 /
Ename(1):SMITH
Ename(2):ALLEN
Ename(3):WARD
Ename(4):JONES
Ename(5):MARTIN
Ename(6):BLAKE
Ename(7):CLARK
Ename(8):SCOTT
Ename(9):KING
Ename(10):TURNER
Ename(11):ADAMS
Ename(12):JAMES
Ename(13):FORD
Ename(14):MILLER
PL/SQL procedure successfully completed.
SQL> |

```

5. Using Index by table type, create a type of numbers indexed by binary integer to store the values from 1 to 10 at continuous index position. Delete the value at position 6 and display all the values.

```
SQL> declare
  2   type indextype is table of Number index by binary_integer;
  3   v1 indextype;
  4   begin
  5     v1(1):=1;
  6     v1(2):=2;
  7     v1(3):=3;
  8     v1(4):=4;
  9     v1(5):=5;
 10    v1(6):=6;
 11    v1(7):=7;
 12    v1(8):=8;
 13    v1(9):=9;
 14    v1(10):=10;
 15   For i in v1.first..v1.last loop
 16     dbms_output.put_line(v1(i));
 17   end loop;
 18   v1.delete(6);
 19   dbms_output.put_line('-----');
 20   For i in 1..5 loop
 21     dbms_output.put_line(v1(i));
 22   end loop;
 23   For i in 7..10 loop
 24     dbms_output.put_line(v1(i));
 25   end loop;
 26 end;
 27 /
```

1
2
3
4
5
6
7
8
9
10

1
2
3
4
5
7
8
9
10

PL/SQL procedure successfully completed.

SQL> |

PRACTICAL NO: -7

1. Using rowtype variable, display the records of the employee whose employee no is given by the user.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> declare
 2   erc emp%rowtype;
 3   begin
 4     select * into erc from emp where empno=&empno;
 5     dbms_output.put_line(erc.ename||' '||erc.sal||' '||erc.job||' '||erc.deptno);
 6   end;
 7 /
Enter value for empno: 7839
old  4: select * into erc from emp where empno=&empno;
new  4: select * into erc from emp where empno=7839;
KING  5000  PRESIDENT  10

PL/SQL procedure successfully completed.
```

2. Using cursor type variable, display the records of the employee.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> declare
 2   cursor c351 is select * from emp;
 3   erc c351%rowtype;
 4   begin
 5     open c351;
 6     loop
 7       fetch c351 into erc;
 8       exit when c351%notfound;
 9       dbms_output.put_line(erc.empno||' '||erc.ename);
10     end loop;
11   end;
12 /
7369  SMITH
7499  ALLEN
7521  WARD
7566  JONES
7654  MARTIN
7698  BLAKE
7782  CLARK
7788  SCOTT
7839  KING
7844  TURNER
7876  ADAMS
7900  JAMES
7902  FORD
7934  MILLER

PL/SQL procedure successfully completed.
```

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> declare
 2   cursor c351 is select * from emp;
 3   erc c351%rowtype;
 4   begin
 5     open c351;
 6     fetch c351 into erc;
 7     while c351%found loop
 8       dbms_output.put_line(erc.empno||' '||erc.ename||' '||erc.deptno);
 9       fetch c351 into erc;
10     end loop;
11   close c351;
12 end;
13 /
7369  SMITH  20
7499  ALLEN  30
7521  WARD   30
7566  JONES  20
7654  MARTIN 30
7698  BLAKE  30
7782  CLARK  10
7788  SCOTT  20
7839  KING   10
7844  TURNER 30
7876  ADAMS  20
7900  JAMES  30
7902  FORD   20
7934  MILLER 10

PL/SQL procedure successfully completed.
```



Oracle SQL*Plus

```
File Edit Search Options Help

SQL> declare
  2  cursor c351 is select * from emp;
  3  begin
  4  for erc in c351 loop
  5    dbms_output.put_line(erc.empno||' '||erc.ename||' '||erc.deptno);
  6  end loop;
  7  end;
  8 /
7369  SMITH  20
7499  ALLEN  30
7521  WARD   30
7566  JONES  20
7654  MARTIN 30
7698  BLAKE  30
7782  CLARK  10
7788  SCOTT  20
7839  KING   10
7844  TURNER 30
7876  ADAMS  20
7900  JAMES  30
7902  FORD   20
7934  MILLER 10

PL/SQL procedure successfully completed.
```

3. Using User based record, create a book type , read and display the values for specific

```
SQL> declare
  2  type book is record
  3  (bid int,bname varchar(25),bprice int);
  4  b1 book;
  5  b2 book;
  6  begin
  7  b1.bid :=351;
  8  b1.bname :='DBHS2';
  9  b1.bprice :=3994;
 10 -- print the details of book1
 11 dbms_output.put_line(b1.bid||' '||b1.bname||' '||b1.bprice);
 12 end;
 13 /
351 DBHS2 3994

PL/SQL procedure successfully completed.

SQL> |
```

PRACTICAL NO: -8

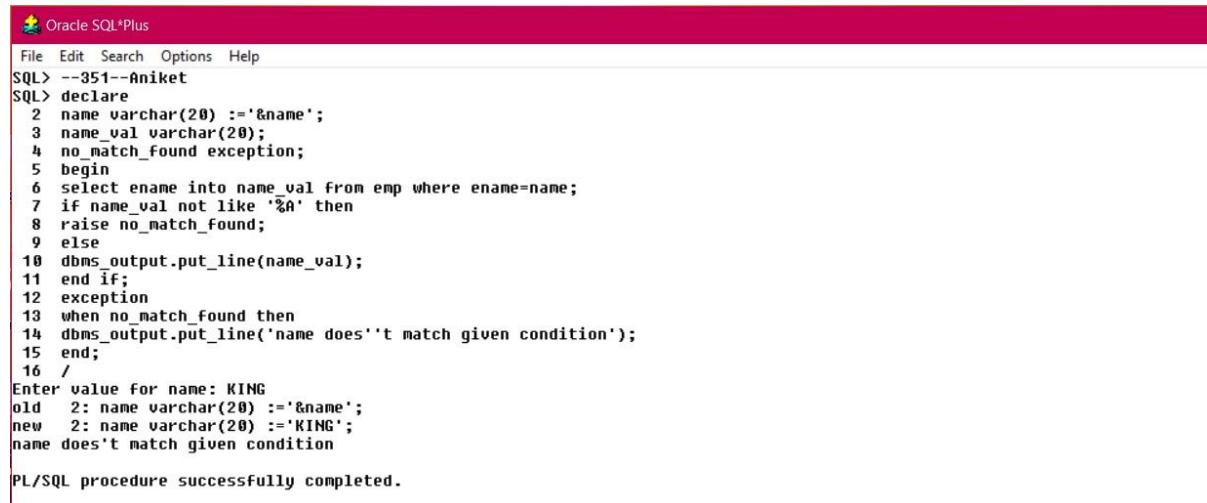
1. Implement System Defined exception.

```

SQL> declare
 2  erec emp%rowtype;
 3  begin
 4
 5    select * into erec from emp where sal>1000;
 6
 7    dbms_output.put_line(erec.sal);
 8  exception
 9  when TOO_MANY_ROWS then
10   dbms_output.put_line('Select statement is fetching than one row');
11  end;
12 /
Select statement is fetching than one row
PL/SQL procedure successfully completed.

```

2. Write a PL/SQL block to raise an exception when an employee name does not start with 'A'.

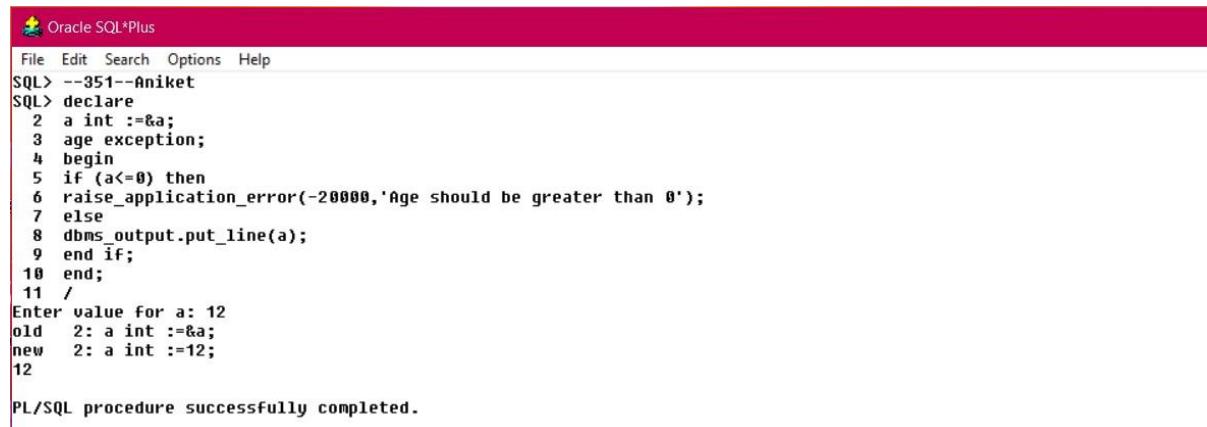


```

Oracle SQL*Plus
File Edit Search Options Help
SQL> --351--Aniket
SQL> declare
 2  name varchar(20) :='&name';
 3  name_val varchar(20);
 4  no_match_found exception;
 5  begin
 6    select ename into name_val from emp where ename=name;
 7    if name_val not like '%A' then
 8      raise no_match_found;
 9    else
10      dbms_output.put_line(name_val);
11    end if;
12  exception
13  when no_match_found then
14    dbms_output.put_line('name doesn't match given condition');
15  end;
16 /
Enter value for name: KING
old  2: name varchar(20) :='&name';
new  2: name varchar(20) :='KING';
name doesn't match given condition
PL/SQL procedure successfully completed.

```

3. Write a PL/SQL block to implement raise_application_error() when the age of a student is entered less than 0.



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> --351--Aniket
SQL> declare
 2  a int :=&a;
 3  age exception;
 4  begin
 5  if (a<=0) then
 6    raise_application_error(-20000,'Age should be greater than 0');
 7  else
 8    dbms_output.put_line(a);
 9  end if;
10 end;
11 /
Enter value for a: 12
old  2: a int :=&a;
new  2: a int :=12;
12
PL/SQL procedure successfully completed.

```

4. Write a PL/SQL block to raise an exception when divisor is 0.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> --351--Aniket
SQL> declare
 2  a int := &a;
 3  b int := &b;
 4  c int;
 5  zerod exception;
 6  begin
 7  if (b=0) then
 8  raise zerod;
 9  else
10  c:=a/b;
11  dbms_output.put_line(c);
12  end if;
13  exception
14  when zerod then
15  dbms_output.put_line('b must be>0');
16  end;
17 /
Enter value for a: 15
old  2: a int := &a;
new  2: a int := 15;
Enter value for b: 0
old  3: b int := &b;
new  3: b int := 0;
b must be>0
PL/SQL procedure successfully completed.
```

5. Write a PL/SQL block to raise an exception when data record is not found in the table.

```
SQL> --351--Aniket
SQL> declare
 2  en varchar(10);
 3  begin
 4  select ename into en from emp where deptno=&deptno;
 5  dbms_output.put_line(en);
 6  exception
 7  when NO_DATA_FOUND then
 8  dbms_output.put_line('dept does not exist');
 9  end;
10 /
Enter value for deptno: 40
old  4: select ename into en from emp where deptno=&deptno;
new  4: select ename into en from emp where deptno=40;
dept does not exist
PL/SQL procedure successfully completed.
SQL> |
```

PRACTICAL NO: - 9

1. Write a PL/SQL block to UPDATE the salary of the employee by 500 whose department number is given by the user and display the number of rows if UPDATED. Otherwise DISPLAY the message ‘No Records Updated’ by using IMPLICIT CURSOR.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> declare
  2   m emp.deptno%type := &deptno;
  3   begin
  4     update emp
  5       set sal=sal+500 where deptno=m;
  6     if sql%found then
  7       dbms_output.put_line('No Rows Found');
  8     else
  9       dbms_output.put_line('No Rows Updated');
 10    end if;
 11  end;
 12 /
Enter value for deptno: 10
old  2: m emp.deptno%type := &deptno;
new  2: m emp.deptno%type := 10;
No Rows Found

PL/SQL procedure successfully completed.

SQL> /
Enter value for deptno: 40
old  2: m emp.deptno%type := &deptno;
new  2: m emp.deptno%type := 40;
No Rows Updated

PL/SQL procedure successfully completed.
```

2. Write a PI/SQL block using cursor to display the employee names and their salaries till the record is found using while loop.

```
SQL> declare
  2   cursor c2 is select * from emp;
  3   erec c2%rowtype;
  4   begin
  5     open c2;
  6     fetch c2 into erec;
  7     while c2%found loop
  8       dbms_output.put_line(erec.ename||' '||erec.sal);
  9     fetch c2 into erec;
 10   end loop;
 11 end;
 12 /
SMITH  800
ALLEN 1600
WARD  1250
JONES  2975
MARTIN 1250
BLAKE  2850
CLARK  2950
SCOTT  3000
KING   5500
TURNER 1500
ADAMS  1100
JAMES  950
FORD   3000
MILLER 1800

PL/SQL procedure successfully completed.

SQL>
```

3. Write a PL/SQL block to DISPLAY the employee name along with their jobs using BASIC LOOP.

```
SQL> declare
 2  erec emp%rowtype;
 3  cursor c3 is
 4  select * from emp;
 5  begin
 6  open c3;
 7  loop
 8  fetch c3 into erec;
 9  dbms_output.put_line(erec.ename||' '||erec.job);
10  exit when c3%notfound;
11 end loop;
12 end;
13 /
SMITH CLERK
ALLEN SALESMAN
WARD SALESMAN
JONES MANAGER
MARTIN SALESMAN
BLAKE MANAGER
CLARK MANAGER
SCOTT ANALYST
KING PRESIDENT
TURNER SALESMAN
ADAMS CLERK
JAMES CLERK
FORD ANALYST
MILLER CLERK
MILLER CLERK

PL/SQL procedure successfully completed.

SQL> |
```

4. Write a PL/SQL block to display the employee names and their jobs using for loop.

```
SQL> declare
 2  cursor c4 is
 3  select * from emp;
 4  begin
 5  for erec in c4 loop
 6  dbms_output.put_line(erec.ename||' '||erec.job);
 7  end loop;
 8  end;
 9 /
SMITH CLERK
ALLEN SALESMAN
WARD SALESMAN
JONES MANAGER
MARTIN SALESMAN
BLAKE MANAGER
CLARK MANAGER
SCOTT ANALYST
KING PRESIDENT
TURNER SALESMAN
ADAMS CLERK
JAMES CLERK
FORD ANALYST
MILLER CLERK

PL/SQL procedure successfully completed.

SQL> |
```

5. Write a PL/SQL block using cursor to display the name and job of employee who are working as ‘MANAGER’. The value of the job as ‘MANAGER’ should be read by the user.

```
SQL> declare
  2   cursor c5(j varchar) is select * from emp where job=j;
  3   j varchar(25);
  4   elec c5%rowtype;
  5   begin
  6     j :='MANAGER';
  7     for elec in c5(j) loop
  8       dbms_output.put_line(elec.ename||' '||elec.job);
  9     end loop;
 10   end;
 11 /
JONES  MANAGER
BLAKE  MANAGER
CLARK  MANAGER

PL/SQL procedure successfully completed.

SQL>
```

6. Write a PL/SQL block to display the details of the employee who are working in department 30 with the help of REF CURSOR.

```
SQL> declare
  2   type employeee is ref cursor return emp%rowtype;
  3   c6 employeee;
  4   erec c6%rowtype;
  5   begin
  6     open c6 for select * from emp where deptno=30;
  7     loop
  8       fetch c6 into erec;
  9       exit when c6%notfound;
 10      dbms_output.put_line(erec.ename||' '||erec.deptno||' '||erec.job);
 11    end loop;
 12  end;
 13 /
ALLEN  30  SALESMAN
WARD   30  SALESMAN
MARTIN 30  SALESMAN
BLAKE  30  MANAGER
TURNER 30  SALESMAN
JAMES   30  CLERK

PL/SQL procedure successfully completed.

SQL> |
```

7. Write a PL/SQL block to check whether the CURSOR is OPEN or not and to DISPLAY the appropriate message using EXPLICIT CURSOR.

```
SQL> declare
  2   cursor c7 is select * from emp;
  3   erec emp%rowtype;
  4   begin
  5     open c7;
  6     if not c7%isopen then
  7       dbms_output.put_line('Cursor is not open.');
  8     end if;
  9     if c7%isopen then
 10       dbms_output.put_line('Cursor is open.');
 11     end if;
 12   end;
 13 /
Cursor is open.

PL/SQL procedure successfully completed.

SQL> |
```

8. Write a PL/SQL block to DISPLAY the employee name and their hire date using FOR LOOP.

```
SQL> declare
 2 cursor c8 is
 3 select * from emp;
 4 begin
 5 for erec in c8 loop
 6 dbms_output.put_line(erec.ename||' '||erec.hiredate);
 7 end loop;
 8 end;
 9 /
SMITH 17-DEC-80
ALLEN 20-FEB-81
WARD 22-FEB-81
JONES 02-APR-81
MARTIN 28-SEP-81
BLAKE 01-MAY-81
CLARK 09-JUN-81
SCOTT 19-APR-87
KING 17-NOV-81
TURNER 08-SEP-81
ADAMS 23-MAY-87
JAMES 03-DEC-81
FORD 03-DEC-81
MILLER 23-JAN-82

PL/SQL procedure successfully completed.
```

SQL> |

9. Write a PL/SQL block to DISPLAY the name of the employee whose name starts with ‘A’ and department is 20, where department number is passed by the user.

```
SQL> declare
 2 cursor c9(d int) is select * from emp where ename like 'A%' and deptno=d;
 3 d int;
 4 erec emp%rowtype;
 5 begin
 6 d :=20;
 7 for erec in c9(d) loop
 8 dbms_output.put_line('Name'||','||'DeptNo');
 9 dbms_output.put_line(erec.ename||','||erec.deptno);
10 exit when c9%notfound;
11 end loop;
12 end;
13 /
Name DeptNo
ADAMS 20

PL/SQL procedure successfully completed.
```

SQL>

10. Write a PL/SQL block to DISPLAY the employee number and employee name who are working as ‘MANAGER’ and earning salary greater than 1000.

```
SQL> declare
 2 cursor c10 is select * from emp where job='MANAGER' and sal>1000;
 3 erec emp%rowtype;
 4 begin
 5 open c10;
 6 loop
 7 fetch c10 into erec;
 8 dbms_output.put_line(erec.empno||' '||erec.ename);
 9 exit when c10%notfound;
10 end loop;
11 end;
12 /
7566 JONES
7698 BLAKE
7782 CLARK
7782 CLARK

PL/SQL procedure successfully completed.
```

SQL> |