PL/SQL

Practical Journal

Submitted by

Rohan Poojary

Roll no:- 31011121037

Department Of Computer Science
Somaiya Vidyavihar University

SK SOMAIYA COLLEGE

Practical no	TOPICS
1.1	Write a PL/SQL code to print the statement “Hello World!”
1.2	Write a PL/SQL code to perform the addition of two numbers
1.3	Write a PL/SQL code to perform addition, subtraction, division and multiplication
1.4	Write a PL/SQL code to calculate the diameter, area, and circumference of a circle using its radius.
2.1	Write a PL/SQL code to check whether an entered number is greater than 20. If it is less than 20 then print the message
2.2	Write a PL/SQL code to check if an entered number is less than or greater than 20 using IF/ELSE statement
2.3	Write a PL/SQL code to check if the entered number is odd or even
2.4	Write a PL/SQL Code to check if the entered number is 10,20,30 or something else and print the value of the number
2.5	Write a PL/SQL Code to display commission according to sales-revenue..
2.6	Write a PL/SQL Code to display message according to the grade..
3.1	Write a pl/sql code to create a table and insert 5 records and then deduct 100 rs fine from the account whose balance is less than the specified balance amount (less than 500)
3.2	Write a pl/sql code to create a table and insert 5 records and give 150 rs bonus from the account whose balance is more than the specified balance amount(more than 500)
4.1	Write a PL/SQL Code to display a number from 1 to 20 using a simple loop and exit statement
4.2	Write a PL/SQL Code to display a number from 1 to 20 using a simple loop and the exit when statement
4.3	Write a PL/SQL Code to display even numbers from 1 to 20 using a while loop

4.4	Write a PL/SQL Code to calculate the area of circle where the radius varies from 2 to 7 using while loop
4.5	Write a PL/SQL Code to calculate the area of circle where the radius varies from 2 to 7 using for loop
5.1	Create a table to insert the values of radius and area and use a for loop to print the area from 2 to 7 and insert the data into the table created
5.2	Write a PL/SQL code to calculate the table of a number
5.3	Write a PL/SQL code to display the number in reverse order from 51 to 100 using for loop
5.4	Write a PL/SQL code to display the reverse of a number
5.5	Write a PL/SQL code to display numbers from 11 to 20 but when the number is 15 it mentions the ‘value of a is 15’ using goto statement
6.1	Write a PL/SQL Code to display message according to the grade..
6.2	Write a PL/SQL code if sales is greater than 200000 then commission is..
7.1	Write a program to create a procedure for displaying ‘ HELLO WORLD! ’ message.
7.2	Write a program to create a procedure for multiplication of two numbers.
7.3	Write a program to create a procedure for displaying a maximum of two numbers.
7.4	Write a program to create procedure inside PL/SQL code to find a factorial of a number.
7.5	Write a program to create a procedure inside PL/SQL code to find a square of a number. Write a program to drop procedure inside PL/SQL.
8.1	Write a PL/SQL code to create a function to find the maximum of two numbers
8.2	Write a PL/SQL code to create a function to find the maximum of two numbers using standalone function

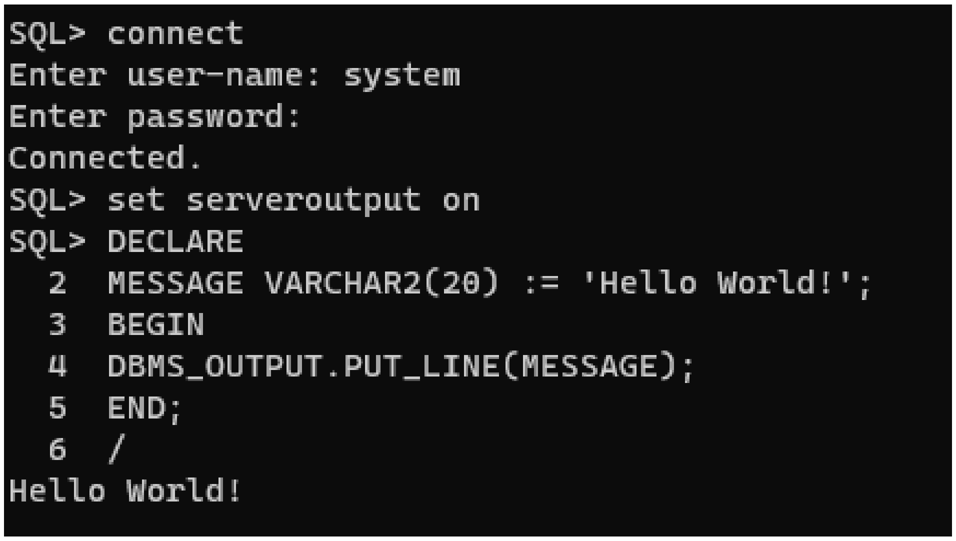
8.3	Create a function in PL/SQL to calculate the average salary of an employee from the table. Create the table employee and insert 5 records into the table.
8.4	Create a function in PL/SQL to calculate the total number of customers from the table. Create the table customers and insert 5 records into the table.
9,1	Create an employee table and insert 2 values into it. Create a trigger that will not allow the user to insert any record into the employee table
9.2	Create a trigger which will not allow the user to insert the record whose id is greater than 100 into the employee table
9.3	Create a trigger that will not allow the user to drop any table
9.4	Write a program in PL/SQL to drop a trigger
10.1	Create a supplier table with attribute suplier_id and name. Create a sequence to increment the id by 2, minvalue 1, maxvalue 99
10.2	Create a student table with attribute student roll number and name. Create a sequence to increment the id by 1, minvalue 1, maxvalue 99
11.1	Create a table employee with attributes employee id, name, address contact number, salary and location.
—--------	—--

Practical 1

1. Write a PL/SQL code to print the statement “Hello World!”

```
SQL> connect
```

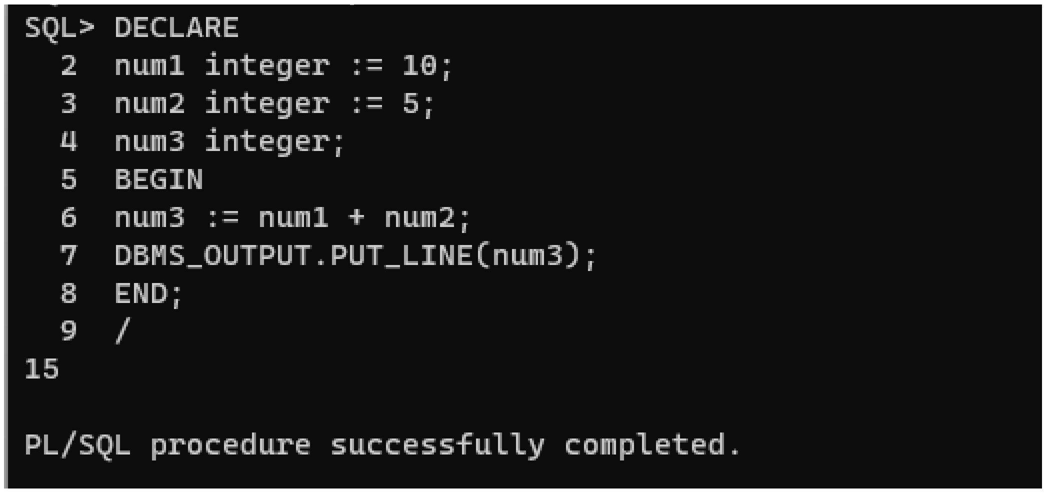
```
Enter user-name: system
Enter password:
Connected.
SQL> set serveroutput on
SQL>
DECLARE
        MESSAGE VARCHAR2(20) := 'Hello World!';
BEGIN
        DBMS_OUTPUT.PUT_LINE(MESSAGE);
END;
 /
Hello World!
```



```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> set serveroutput on
SQL> DECLARE
  MESSAGE VARCHAR2(20) := 'Hello World!';
  BEGIN
  DBMS_OUTPUT.PUT_LINE(MESSAGE);
  END;
  /
Hello World!
```

2. Write a PL/SQL code to perform the addition of two numbers

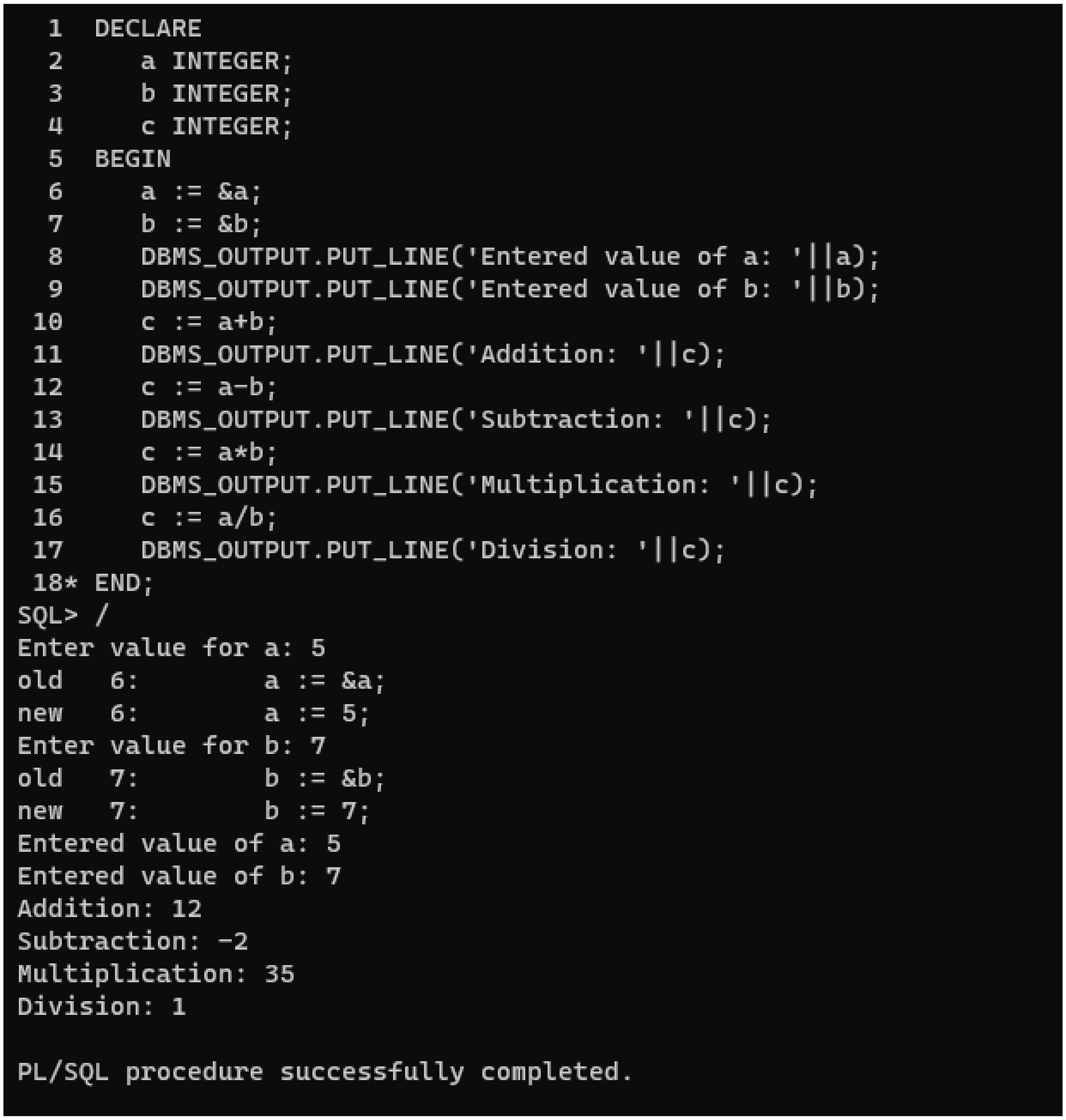
```
DECLARE
        num1 integer := 10;
        num2 integer := 5;
        num3 integer;
BEGIN
        num3 := num1 + num2;
        DBMS_OUTPUT.PUT_LINE(num3);
END;
/
```



```
SQL> DECLARE
     num1 integer := 10;
     num2 integer := 5;
     num3 integer;
     BEGIN
     num3 := num1 + num2;
     DBMS_OUTPUT.PUT_LINE(num3);
     END;
     /
15

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL code to perform addition, subtraction, division and multiplication

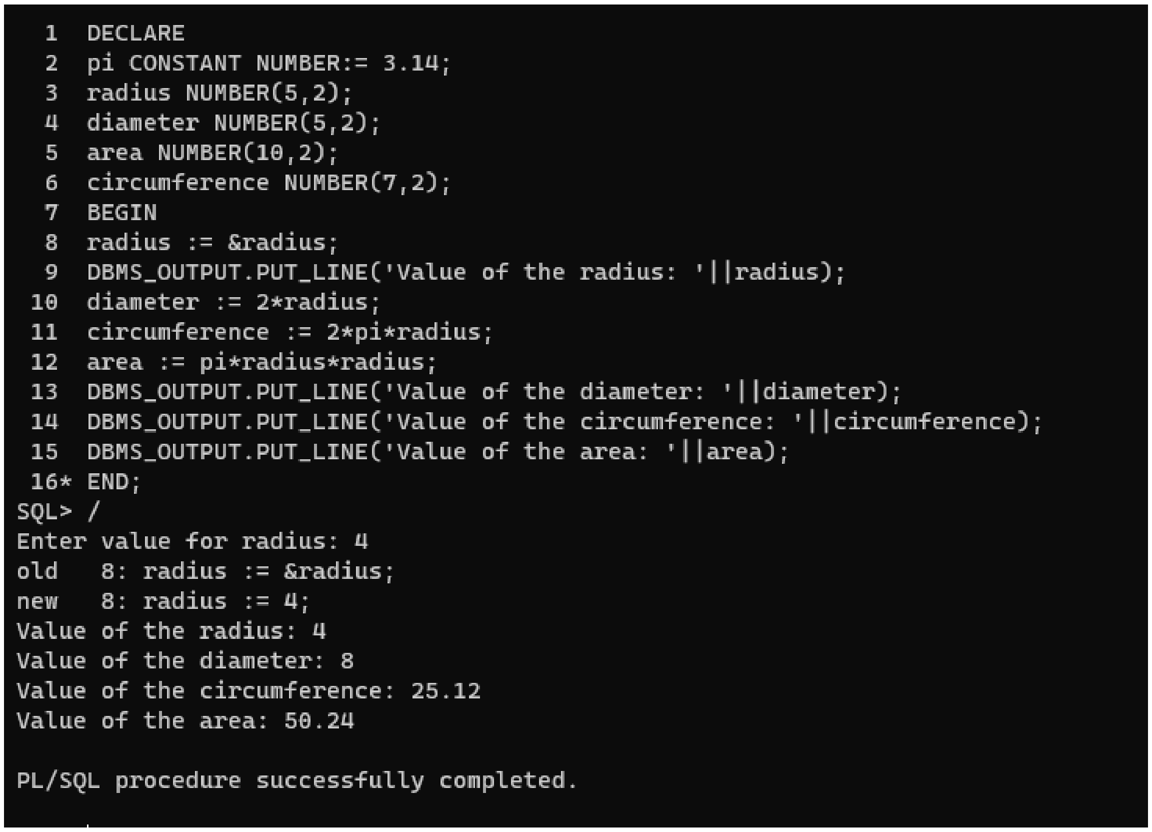
```
DECLARE
        a INTEGER;
        b INTEGER;
        c INTEGER;
BEGIN
        a := &a;
        b := &b;
        DBMS_OUTPUT.PUT_LINE('Entered value of a: '||a);
        DBMS_OUTPUT.PUT_LINE('Entered value of b: '||b);
        c := a+b;
        DBMS_OUTPUT.PUT_LINE('Addition: '||c);
        c := a-b;
        DBMS_OUTPUT.PUT_LINE('Subtraction: '||c);
        c := a*b;
        DBMS_OUTPUT.PUT_LINE('Multiplication: '||c);
        c := a/b;
        DBMS_OUTPUT.PUT_LINE('Division: '||c);
END;
/
```



```
  1  DECLARE
  2     a INTEGER;
  3     b INTEGER;
  4     c INTEGER;
  5  BEGIN
  6     a := &a;
  7     b := &b;
  8     DBMS_OUTPUT.PUT_LINE('Entered value of a: '||a);
  9     DBMS_OUTPUT.PUT_LINE('Entered value of b: '||b);
 10     c := a+b;
 11     DBMS_OUTPUT.PUT_LINE('Addition: '||c);
 12     c := a-b;
 13     DBMS_OUTPUT.PUT_LINE('Subtraction: '||c);
 14     c := a*b;
 15     DBMS_OUTPUT.PUT_LINE('Multiplication: '||c);
 16     c := a/b;
 17     DBMS_OUTPUT.PUT_LINE('Division: '||c);
 18* END;
SQL> /
Enter value for a: 5
old   6:        a := &a;
new   6:        a := 5;
Enter value for b: 7
old   7:        b := &b;
new   7:        b := 7;
Entered value of a: 5
Entered value of b: 7
Addition: 12
Subtraction: -2
Multiplication: 35
Division: 1

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL code to calculate the diameter, area, and circumference of a circle using its radius.

```
DECLARE
        pi CONSTANT NUMBER:= 3.14;
        radius NUMBER(5,2);
        diameter NUMBER(5,2);
        area NUMBER(10,2);
        circumference NUMBER(7,2);
BEGIN
        radius := &radius;
        DBMS_OUTPUT.PUT_LINE('Value of the radius: '||radius);
        diameter := 2*radius;
        circumference := 2*pi*radius;
        area := pi*radius*radius;
        DBMS_OUTPUT.PUT_LINE('Value of the diameter: '||diameter);
        DBMS_OUTPUT.PUT_LINE('Value of the circumference: '||circumference);
        DBMS_OUTPUT.PUT_LINE('Value of the area: '||area);
END;
```

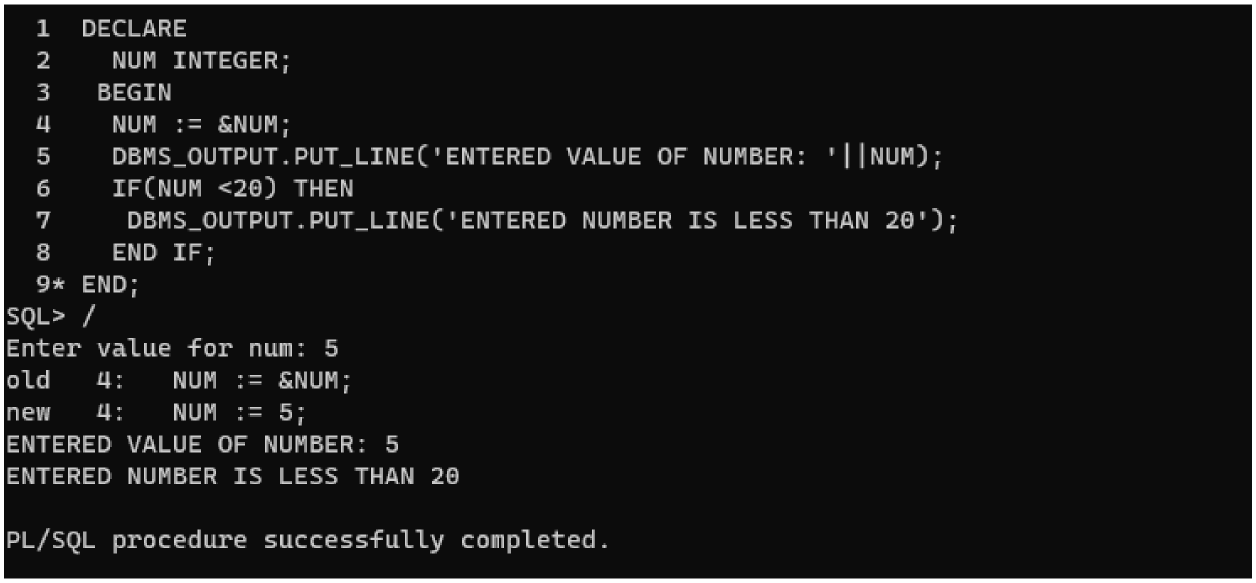


```
DECLARE
pi CONSTANT NUMBER:= 3.14;
radius NUMBER(5,2);
diameter NUMBER(5,2);
area NUMBER(10,2);
circumference NUMBER(7,2);
BEGIN
radius := &radius;
DBMS_OUTPUT.PUT_LINE('Value of the radius: '||radius);
diameter := 2*radius;
circumference := 2*pi*radius;
area := pi*radius*radius;
DBMS_OUTPUT.PUT_LINE('Value of the diameter: '||diameter);
DBMS_OUTPUT.PUT_LINE('Value of the circumference: '||circumference);
DBMS_OUTPUT.PUT_LINE('Value of the area: '||area);
END;
SQL> /
Enter value for radius: 4
old   8: radius := &radius;
new   8: radius := 4;
Value of the radius: 4
Value of the diameter: 8
Value of the circumference: 25.12
Value of the area: 50.24

PL/SQL procedure successfully completed.
```

Practical 2

1. Write a PL/SQL code to check whether an entered number is greater than 20. If it is less than 20 then print the message.

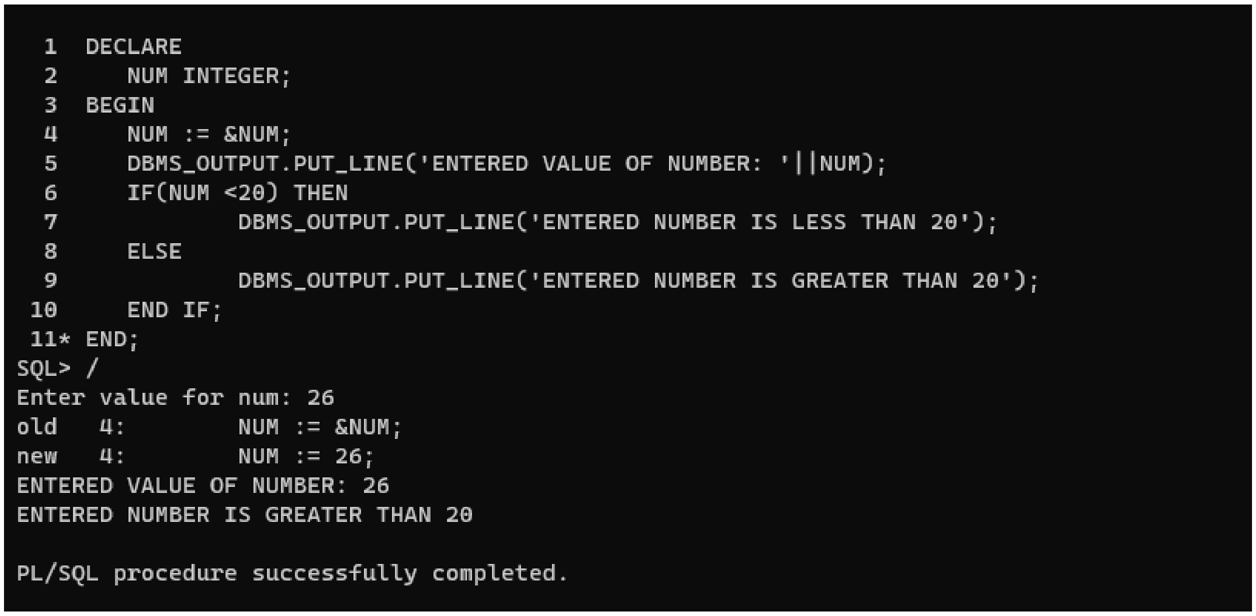
```
DECLARE
        NUM INTEGER;
 BEGIN
        NUM := &NUM;
        DBMS_OUTPUT.PUT_LINE('ENTERED VALUE OF NUMBER: '||NUM);
        IF(NUM <20) THEN
                DBMS_OUTPUT.PUT_LINE('ENTERED NUMBER IS LESS THAN 20');
         END IF;
END;
/
```



```
DECLARE
  NUM INTEGER;
 BEGIN
  NUM := &NUM;
  DBMS_OUTPUT.PUT_LINE('ENTERED VALUE OF NUMBER: '||NUM);
  IF(NUM <20) THEN
   DBMS_OUTPUT.PUT_LINE('ENTERED NUMBER IS LESS THAN 20');
  END IF;
END;
SQL> /
Enter value for num: 5
old   4:   NUM := &NUM;
new   4:   NUM := 5;
ENTERED VALUE OF NUMBER: 5
ENTERED NUMBER IS LESS THAN 20

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL code to check if an entered number is less than or greater than 20 using IF/ELSE statement

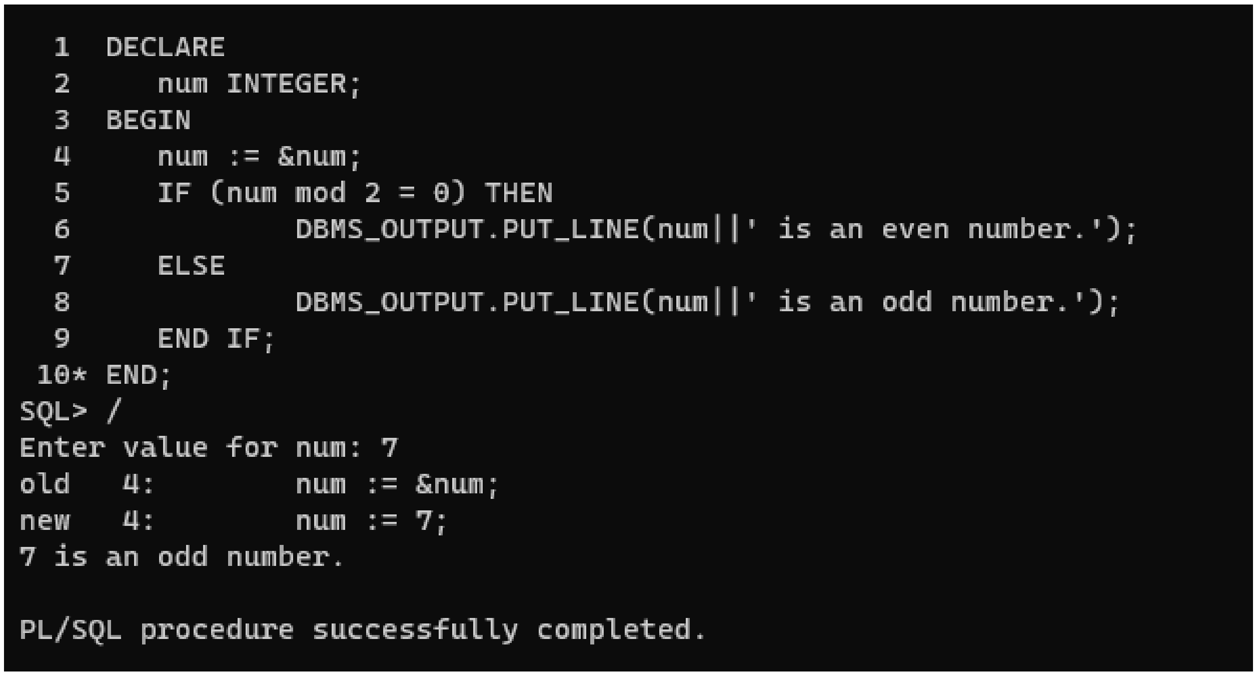
```
DECLARE
        NUM INTEGER;
BEGIN
        NUM := &NUM;
        DBMS_OUTPUT.PUT_LINE('ENTERED VALUE OF NUMBER: '||NUM);
        IF(NUM <20) THEN
                DBMS_OUTPUT.PUT_LINE('ENTERED NUMBER IS LESS THAN 20');
        ELSE
                DBMS_OUTPUT.PUT_LINE('ENTERED NUMBER IS GREATER THAN 20');
        END IF;
END;
/
```



```
DECLARE
   NUM INTEGER;
BEGIN
   NUM := &NUM;
   DBMS_OUTPUT.PUT_LINE('ENTERED VALUE OF NUMBER: '||NUM);
   IF(NUM <20) THEN
           DBMS_OUTPUT.PUT_LINE('ENTERED NUMBER IS LESS THAN 20');
   ELSE
           DBMS_OUTPUT.PUT_LINE('ENTERED NUMBER IS GREATER THAN 20');
   END IF;
END;
SQL> /
Enter value for num: 26
old   4:        NUM := &NUM;
new   4:        NUM := 26;
ENTERED VALUE OF NUMBER: 26
ENTERED NUMBER IS GREATER THAN 20

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL code to check if the entered number is odd or even

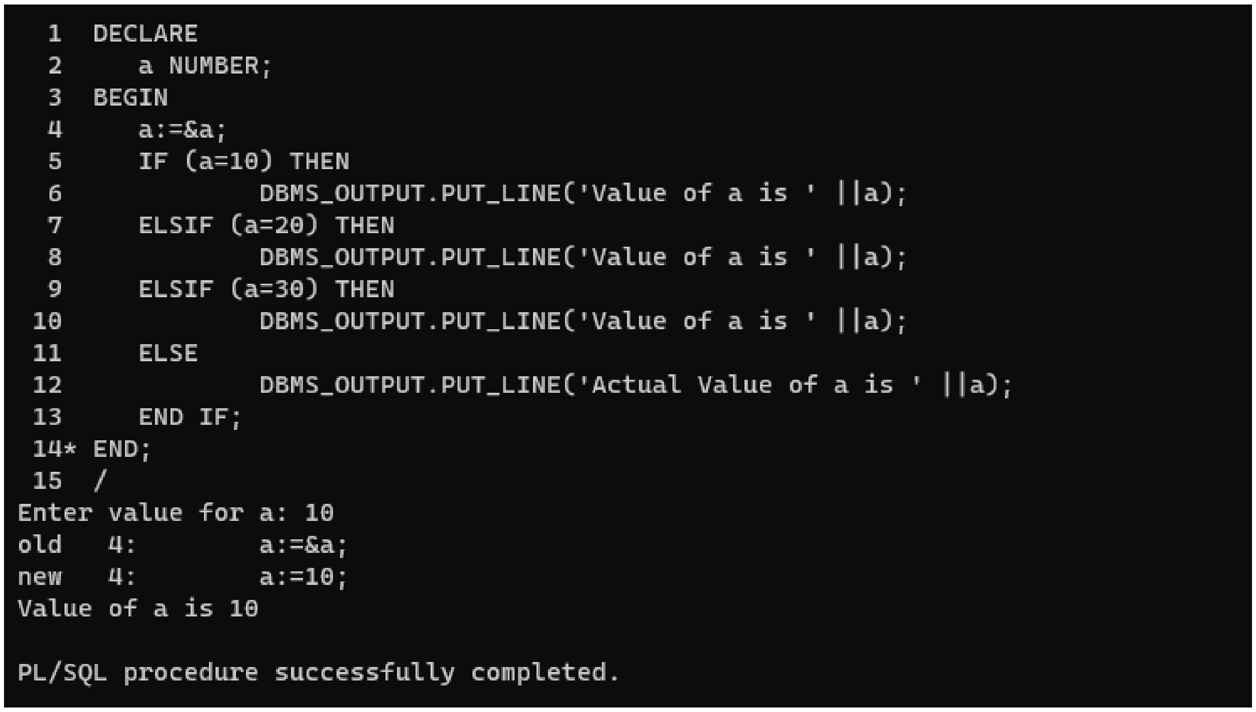
```
DECLARE
        num INTEGER;
BEGIN
        num := &num;
        IF (num mod 2 = 0) THEN
                DBMS_OUTPUT.PUT_LINE(num||' is an even number.');
        ELSE
                DBMS_OUTPUT.PUT_LINE(num||' is an odd number.');
        END IF;
END;
/
```



```
DECLARE
   num INTEGER;
BEGIN
   num := &num;
   IF (num mod 2 = 0) THEN
           DBMS_OUTPUT.PUT_LINE(num||' is an even number.');
   ELSE
           DBMS_OUTPUT.PUT_LINE(num||' is an odd number.');
   END IF;
END;
SQL> /
Enter value for num: 7
old   4:        num := &num;
new   4:        num := 7;
7 is an odd number.

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL Code to check if the entered number is 10,20,30 or something else and print the value of the number

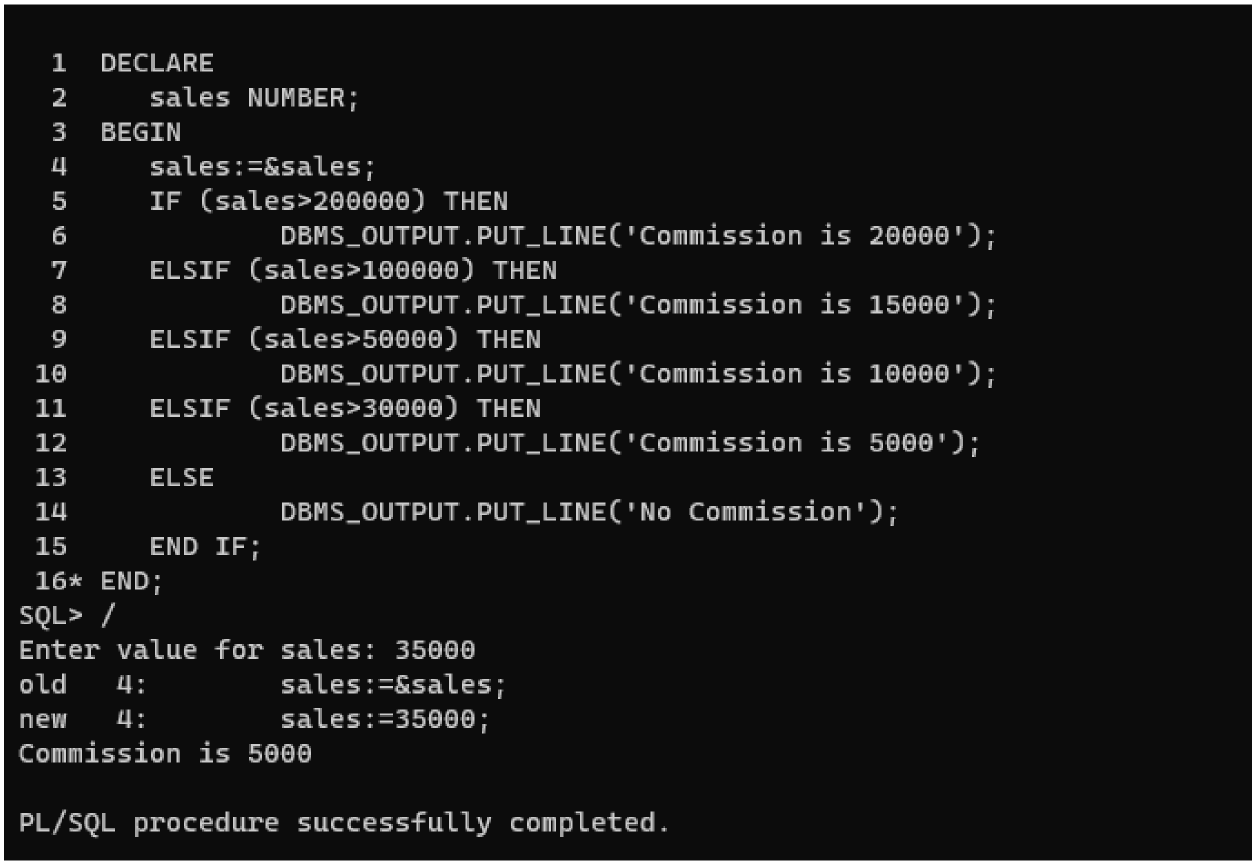
```
DECLARE
        a NUMBER;
BEGIN
        a:=&a;
        IF (a=10) THEN
                DBMS_OUTPUT.PUT_LINE('Value of a is ' ||a);
        ELSIF (a=20) THEN
                DBMS_OUTPUT.PUT_LINE('Value of a is ' ||a);
        ELSIF (a=30) THEN
                DBMS_OUTPUT.PUT_LINE('Value of a is ' ||a);
        ELSE
                DBMS_OUTPUT.PUT_LINE('Actual Value of a is ' ||a);
        END IF;
END;
/
```



```
DECLARE
   a NUMBER;
BEGIN
   a:=&a;
   IF (a=10) THEN
           DBMS_OUTPUT.PUT_LINE('Value of a is ' ||a);
   ELSIF (a=20) THEN
           DBMS_OUTPUT.PUT_LINE('Value of a is ' ||a);
   ELSIF (a=30) THEN
           DBMS_OUTPUT.PUT_LINE('Value of a is ' ||a);
   ELSE
           DBMS_OUTPUT.PUT_LINE('Actual Value of a is ' ||a);
   END IF;
END;
/
Enter value for a: 10
old   4:        a:=&a;
new   4:        a:=10;
Value of a is 10

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL Code to display commission according to sales-revenue
If sales-revenue is greater than 2 lakhs then commission is 20000
If sales-revenue is greater than 1 lakhs then commission is 15000
If sales-revenue is greater than 50000 then commission is 10000
If sales-revenue is greater than 30000 then commission is 5000
Otherwise, don’t give commission.

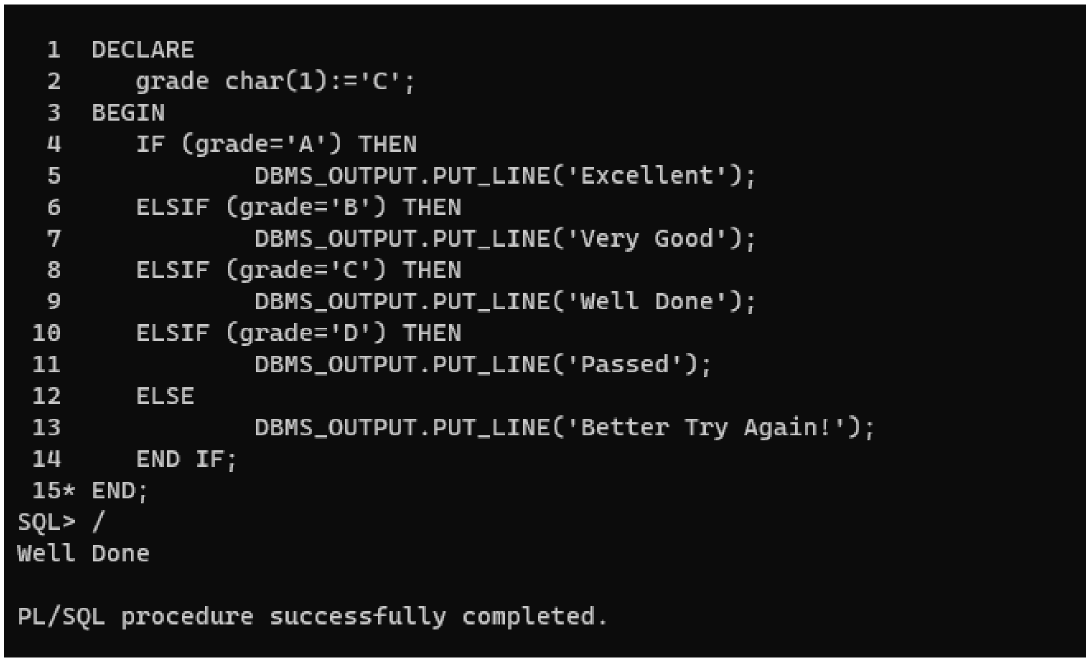
```
DECLARE
        sales NUMBER;
BEGIN
        sales:=&sales;
        IF (sales>200000) THEN
                DBMS_OUTPUT.PUT_LINE('Commission is 20000');
        ELSIF (sales>100000) THEN
                DBMS_OUTPUT.PUT_LINE('Commission is 15000');
        ELSIF (sales>50000) THEN
                DBMS_OUTPUT.PUT_LINE('Commission is 10000');
        ELSIF (sales>30000) THEN
                DBMS_OUTPUT.PUT_LINE('Commission is 5000');
        ELSE
                DBMS_OUTPUT.PUT_LINE('No Commission');
        END IF;
END;
/
```



```
  1  DECLARE
  2     sales NUMBER;
  3  BEGIN
  4     sales:=&sales;
  5     IF (sales>200000) THEN
  6             DBMS_OUTPUT.PUT_LINE('Commission is 20000');
  7     ELSIF (sales>100000) THEN
  8             DBMS_OUTPUT.PUT_LINE('Commission is 15000');
  9     ELSIF (sales>50000) THEN
 10             DBMS_OUTPUT.PUT_LINE('Commission is 10000');
 11     ELSIF (sales>30000) THEN
 12             DBMS_OUTPUT.PUT_LINE('Commission is 5000');
 13     ELSE
 14             DBMS_OUTPUT.PUT_LINE('No Commission');
 15     END IF;
 16* END;
SQL> /
Enter value for sales: 35000
old   4:        sales:=&sales;
new   4:        sales:=35000;
Commission is 5000

PL/SQL procedure successfully completed.
```

6. **Write a PL/SQL Code to display message according to the grade**
If A, then display Excellent
If B, then display Very Good
If C, then display Well Done
If D, then display Passed
If E, then display Better try again

```
DECLARE
        grade char(1):='C';
BEGIN
        IF (grade='A') THEN
                DBMS_OUTPUT.PUT_LINE('Excellent');
        ELSIF (grade='B') THEN
                DBMS_OUTPUT.PUT_LINE('Very Good');
        ELSIF (grade='C') THEN
                DBMS_OUTPUT.PUT_LINE('Well Done');
        ELSIF (grade='D') THEN
                DBMS_OUTPUT.PUT_LINE('Passed');
        ELSE
                DBMS_OUTPUT.PUT_LINE('Better Try Again!');
        END IF;
END;
/
```



```
DECLARE
   grade char(1):='C';
BEGIN
   IF (grade='A') THEN
           DBMS_OUTPUT.PUT_LINE('Excellent');
   ELSIF (grade='B') THEN
           DBMS_OUTPUT.PUT_LINE('Very Good');
   ELSIF (grade='C') THEN
           DBMS_OUTPUT.PUT_LINE('Well Done');
   ELSIF (grade='D') THEN
           DBMS_OUTPUT.PUT_LINE('Passed');
   ELSE
           DBMS_OUTPUT.PUT_LINE('Better Try Again!');
   END IF;
END;
SQL> /
Well Done

PL/SQL procedure successfully completed.
```

Practical 3

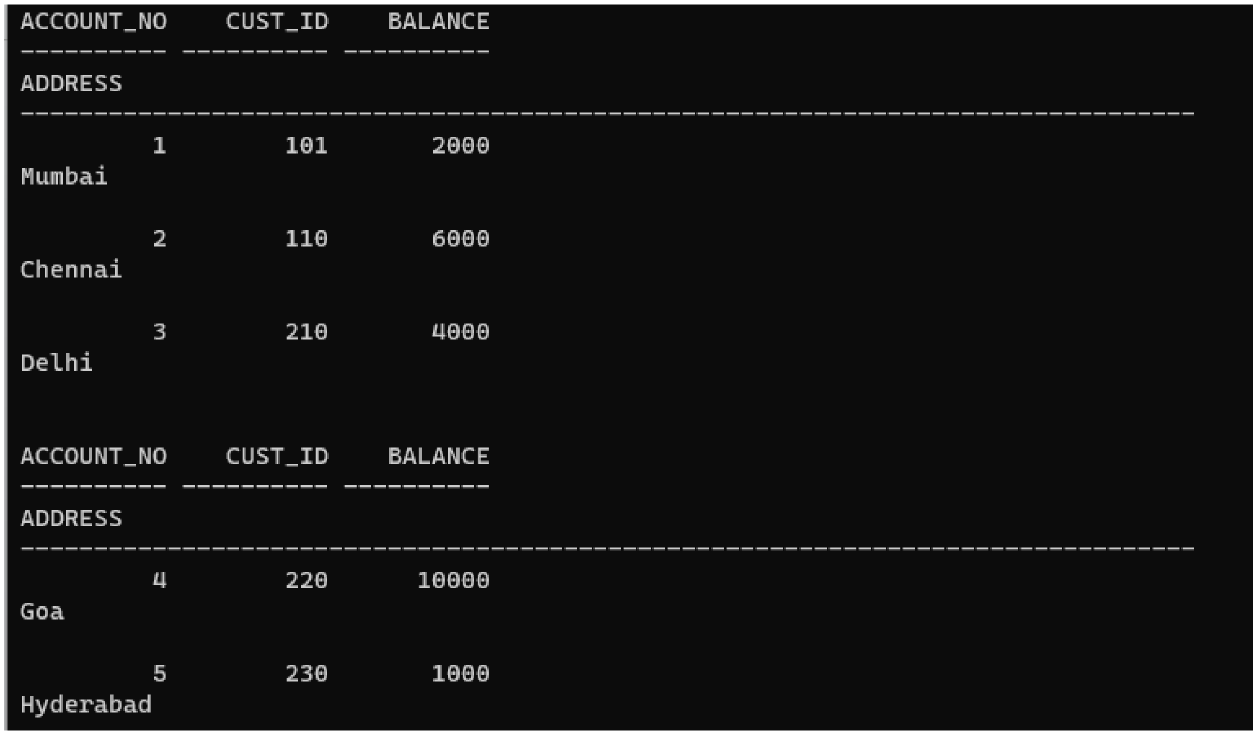
Q1) Write a pl/sql code to create a table and insert 5 records and then deduct 100 rs fine from the account whose balance is less than the specified balance amount (less than 500)

Code:

```
create table Acct_mstr(
 account_no INT,
cust_id int Primary key,
 balance Int,
 Address varchar(200));
```

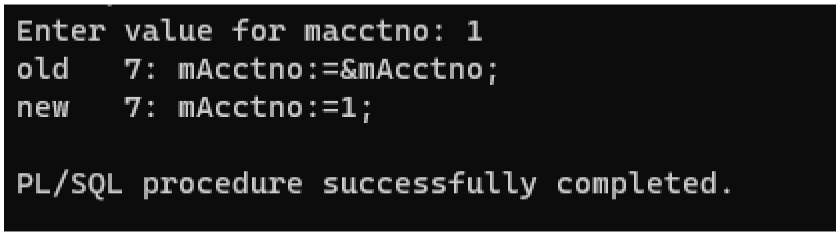
```
Table created.
```

```
insert into Acct_mstr values(1,101,2000,'Mumbai');
1 row created.

insert into Acct_mstr values(2,110,6000,'Chennai');
1 row created.

insert into Acct_mstr values(3,210,4000,'Delhi');
1 row created.

 insert into Acct_mstr values(4,220,10000,'Goa');
1 row created.

 insert into Acct_mstr values(5,230,1000,'Hyderabad');
1 row created.

select * from Acct_mstr;
```



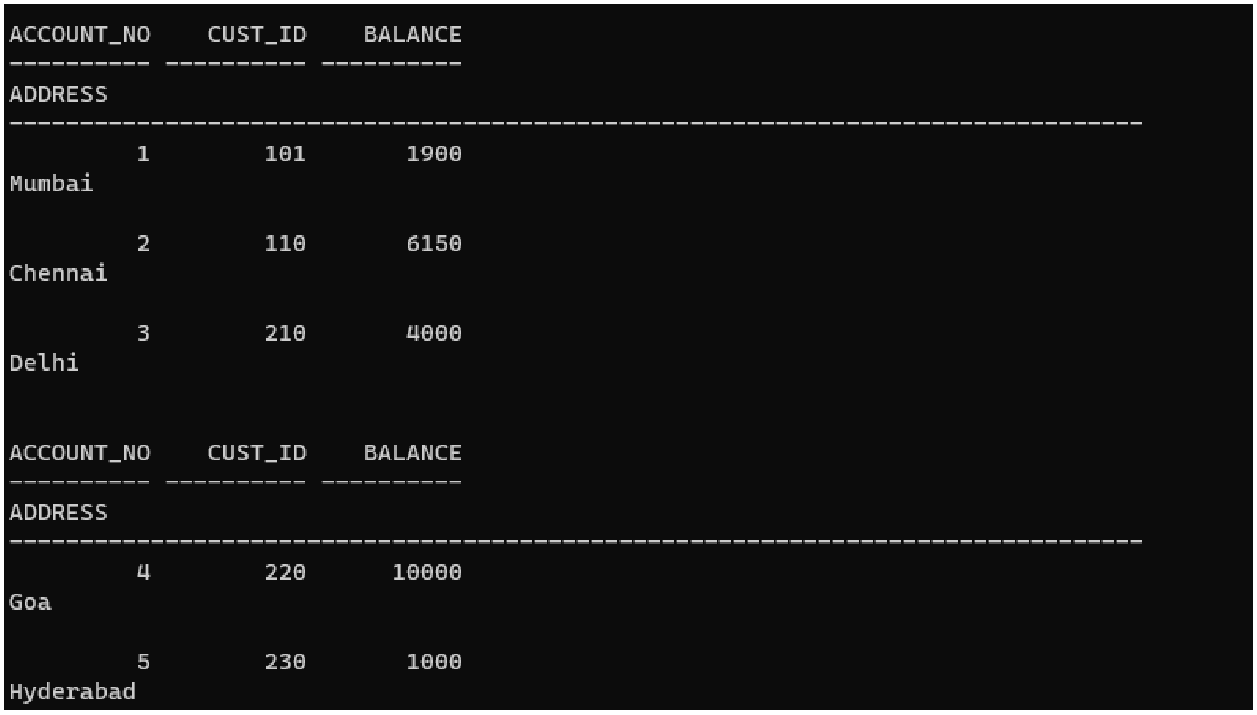
```
ACCOUNT_NO    CUST_ID    BALANCE
---------- ---------- ----------
ADDRESS
--------------------------------------------------------------------------------
         1        101       2000
Mumbai

         2        110       6000
Chennai

         3        210       4000
Delhi


ACCOUNT_NO    CUST_ID    BALANCE
---------- ---------- ----------
ADDRESS
--------------------------------------------------------------------------------
         4        220      10000
Goa

         5        230       1000
Hyderabad
```

```
declare
mcurr_bal number;
mAcctno number;
mfine number:=100;
mMin_bal constant number(6,2):=5000;
begin
mAcctno:=&mAcctno;
select balance into mcurr_bal from Acct_mstr where account_no=mAcctno;
if mcurr_bal<mMin_bal then update Acct_mstr set balance=balance-mfine where
account_no=mAcctno;
end if;
end;
/
```



```
Enter value for macctno: 1
old   7: mAcctno:=&mAcctno;
new   7: mAcctno:=1;

PL/SQL procedure successfully completed.
```

```
select * from Acct_mstr;
```



```
ACCOUNT_NO    CUST_ID    BALANCE
---------- ---------- ----------
ADDRESS
--------------------------------------------------------------------------------
         1        101       1900
Mumbai

         2        110       6150
Chennai

         3        210       4000
Delhi


ACCOUNT_NO    CUST_ID    BALANCE
---------- ---------- ----------
ADDRESS
--------------------------------------------------------------------------------
         4        220      10000
Goa

         5        230       1000
Hyderabad
```

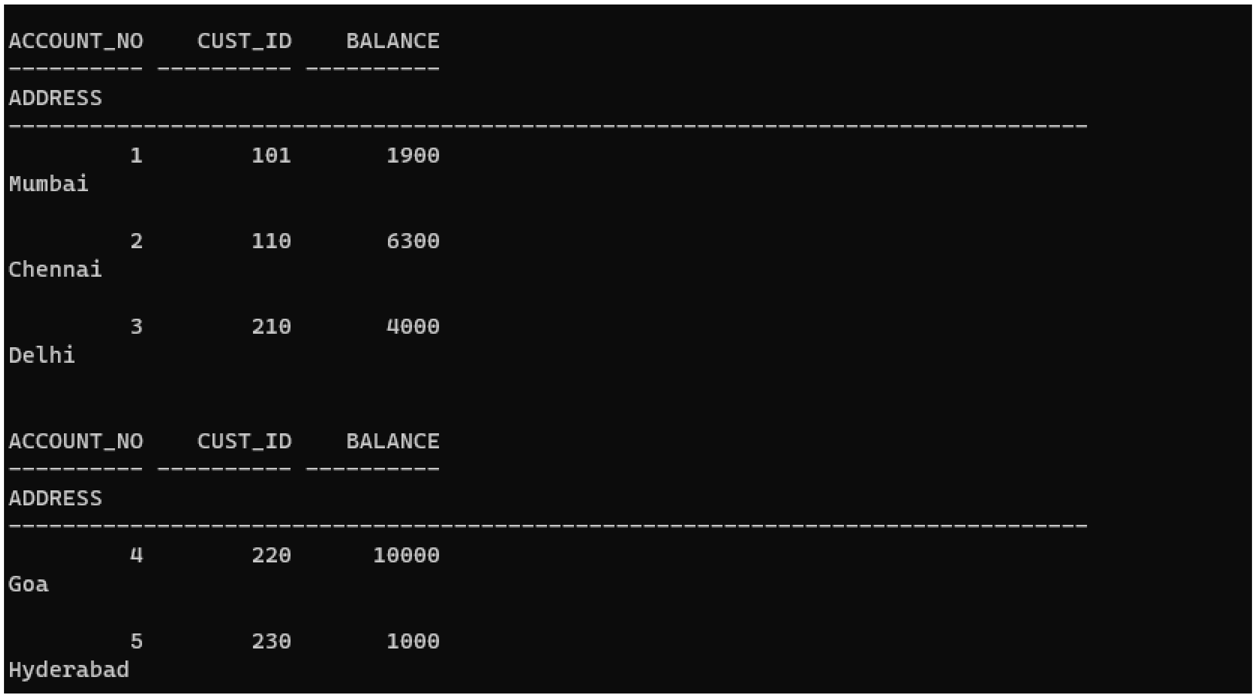
Q2)Write a pl/sql code to create a table and insert 5 records and give 150 rs bonus from the account whose balance is more than the specified balance amount(more than 500)

Code:

```
declare
mcurr_bal number;
mAcctno number;
mfine number:=100;
mBonus number:= 150;
mMin_bal constant number(6,2):=5000;
begin
mAcctno:=&mAcctno;
select balance into mcurr_bal from Acct_mstr where account_no=mAcctno;
if mcurr_bal<mMin_bal then update Acct_mstr set balance=balance-mfine where
account_no=mAcctno;
else update Acct_mstr set balance=balance+mBonus where
account_no=mAcctno;
```

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

```
select * from Acct_mstr;
```

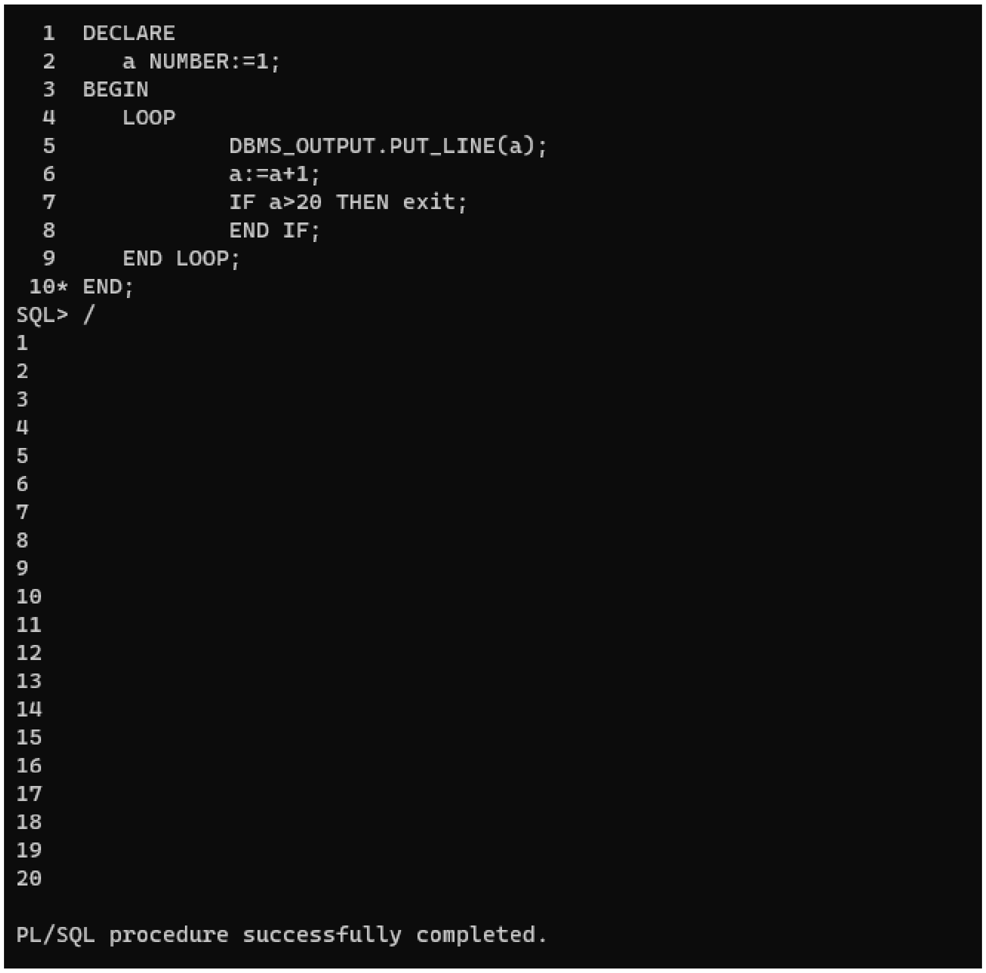


```
ACCOUNT_NO    CUST_ID    BALANCE
---------- ---------- ----------
ADDRESS
--------------------------------------------------------------------------------
         1        101       1900
Mumbai

         2        110       6300
Chennai

         3        210       4000
Delhi


ACCOUNT_NO    CUST_ID    BALANCE
---------- ---------- ----------
ADDRESS
--------------------------------------------------------------------------------
         4        220      10000
Goa

         5        230       1000
Hyderabad
```

Practical 4

1. Write a PL/SQL Code to display a number from 1 to 20 using a simple loop and exit statement

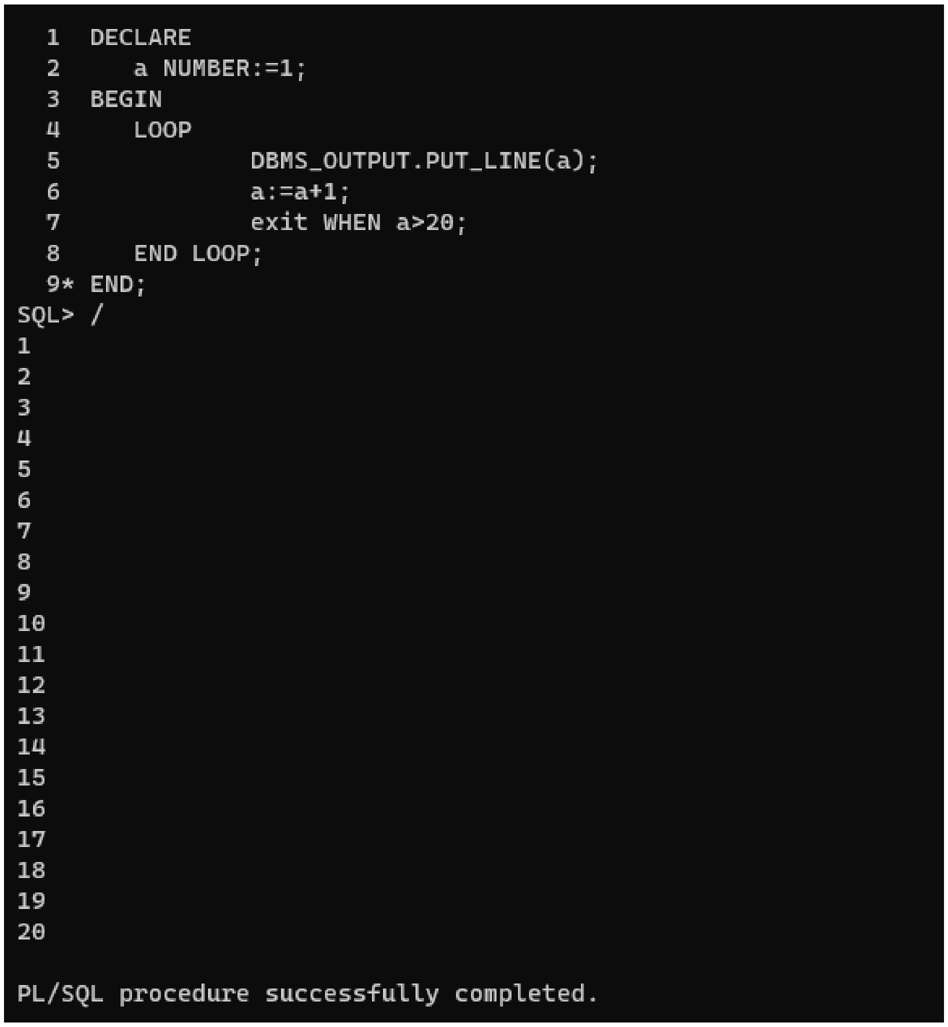
```
DECLARE
        a NUMBER:=1;
BEGIN
        LOOP
                DBMS_OUTPUT.PUT_LINE(a);
                a:=a+1;
                IF a>20 THEN exit;
                END IF;
        END LOOP;
END;
/
```



```
DECLARE
   a NUMBER:=1;
BEGIN
   LOOP
           DBMS_OUTPUT.PUT_LINE(a);
           a:=a+1;
           IF a>20 THEN exit;
           END IF;
   END LOOP;
END;
SQL> /
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL Code to display a number from 1 to 20 using a simple loop and the exit when statement

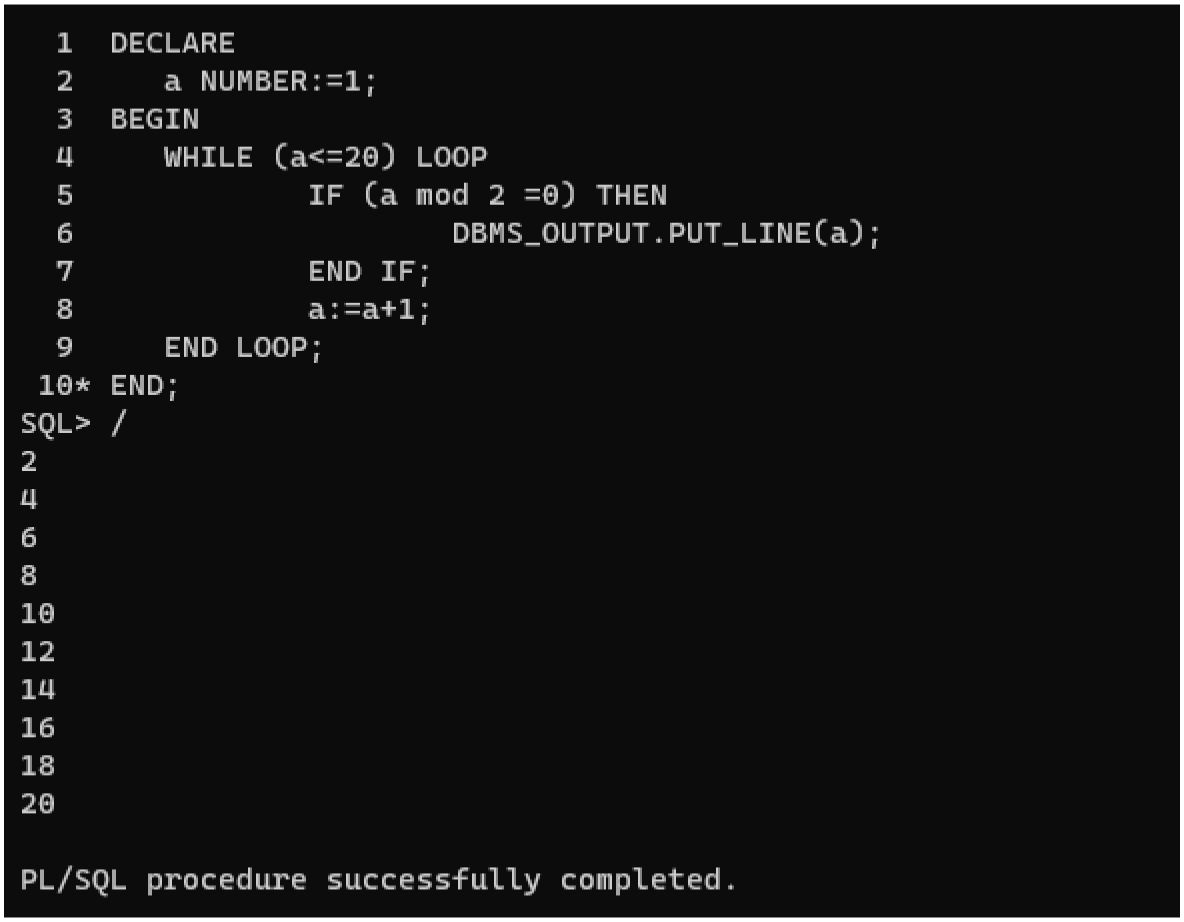
```
DECLARE
        a NUMBER:=1;
BEGIN
        LOOP
                DBMS_OUTPUT.PUT_LINE(a);
                a:=a+1;
                exit WHEN a>20;
        END LOOP;
END;
/
```



```
DECLARE
   a NUMBER:=1;
BEGIN
   LOOP
           DBMS_OUTPUT.PUT_LINE(a);
           a:=a+1;
           exit WHEN a>20;
   END LOOP;
END;
SQL> /
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL Code to display even numbers from 1 to 20 using a while loop

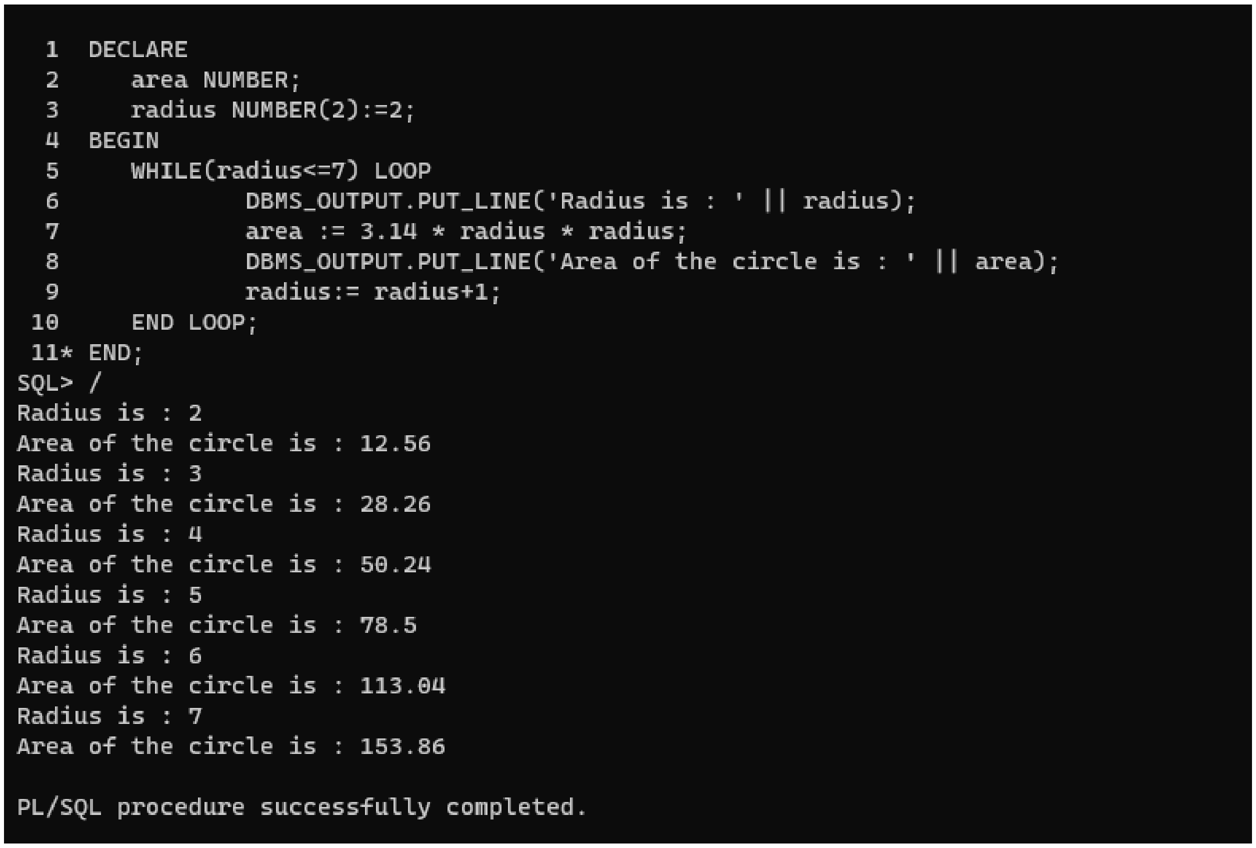
```
DECLARE
        a NUMBER:=1;
BEGIN
        WHILE (a<=20) LOOP
                IF (a mod 2 =0) THEN
                        DBMS_OUTPUT.PUT_LINE(a);
                END IF;
                a:=a+1;
        END LOOP;
END;
/
```



```
DECLARE
   a NUMBER:=1;
BEGIN
   WHILE (a<=20) LOOP
           IF (a mod 2 =0) THEN
                   DBMS_OUTPUT.PUT_LINE(a);
           END IF;
           a:=a+1;
   END LOOP;
END;
SQL> /
2
4
6
8
10
12
14
16
18
20

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL Code to calculate the area of circle where the radius varies from 2 to 7 using while loop

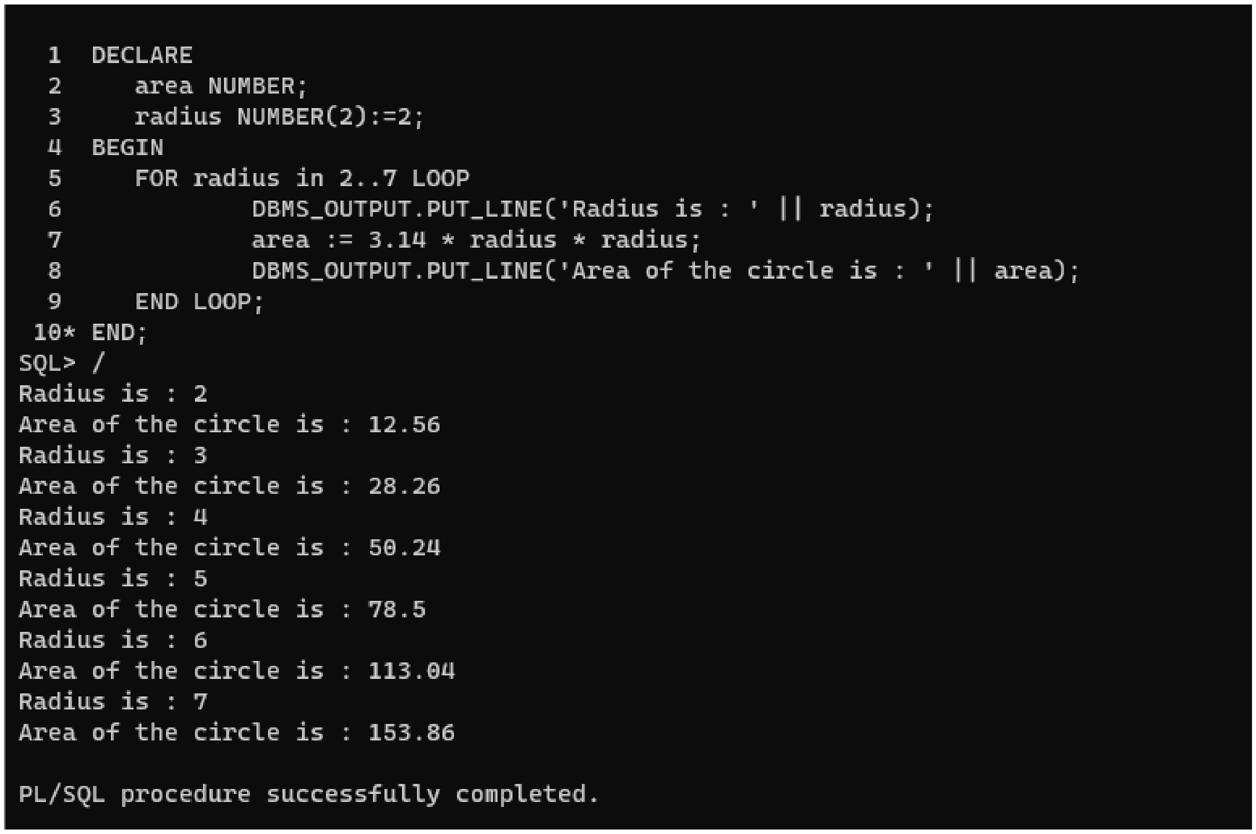
```
DECLARE
        area NUMBER;
        radius NUMBER(2):=2;
BEGIN
        WHILE(radius<=7) LOOP
                DBMS_OUTPUT.PUT_LINE('Radius is : ' || radius);
                area := 3.14 * radius * radius;
                DBMS_OUTPUT.PUT_LINE('Area of the circle is : ' || area);
                radius:= radius+1;
        END LOOP;
END;
/
```



```
DECLARE
   area NUMBER;
   radius NUMBER(2):=2;
BEGIN
   WHILE(radius<=7) LOOP
           DBMS_OUTPUT.PUT_LINE('Radius is : ' || radius);
           area := 3.14 * radius * radius;
           DBMS_OUTPUT.PUT_LINE('Area of the circle is : ' || area);
           radius:= radius+1;
   END LOOP;
END;
SQL> /
Radius is : 2
Area of the circle is : 12.56
Radius is : 3
Area of the circle is : 28.26
Radius is : 4
Area of the circle is : 50.24
Radius is : 5
Area of the circle is : 78.5
Radius is : 6
Area of the circle is : 113.04
Radius is : 7
Area of the circle is : 153.86

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL Code to calculate the area of circle where the radius varies from 2 to 7 using for loop

```
DECLARE
        area NUMBER;
        radius NUMBER(2):=2;
BEGIN
        FOR radius in 2..7 LOOP
                DBMS_OUTPUT.PUT_LINE('Radius is : ' || radius);
                area := 3.14 * radius * radius;
                DBMS_OUTPUT.PUT_LINE('Area of the circle is : ' || area);
        END LOOP;
END;
/
```

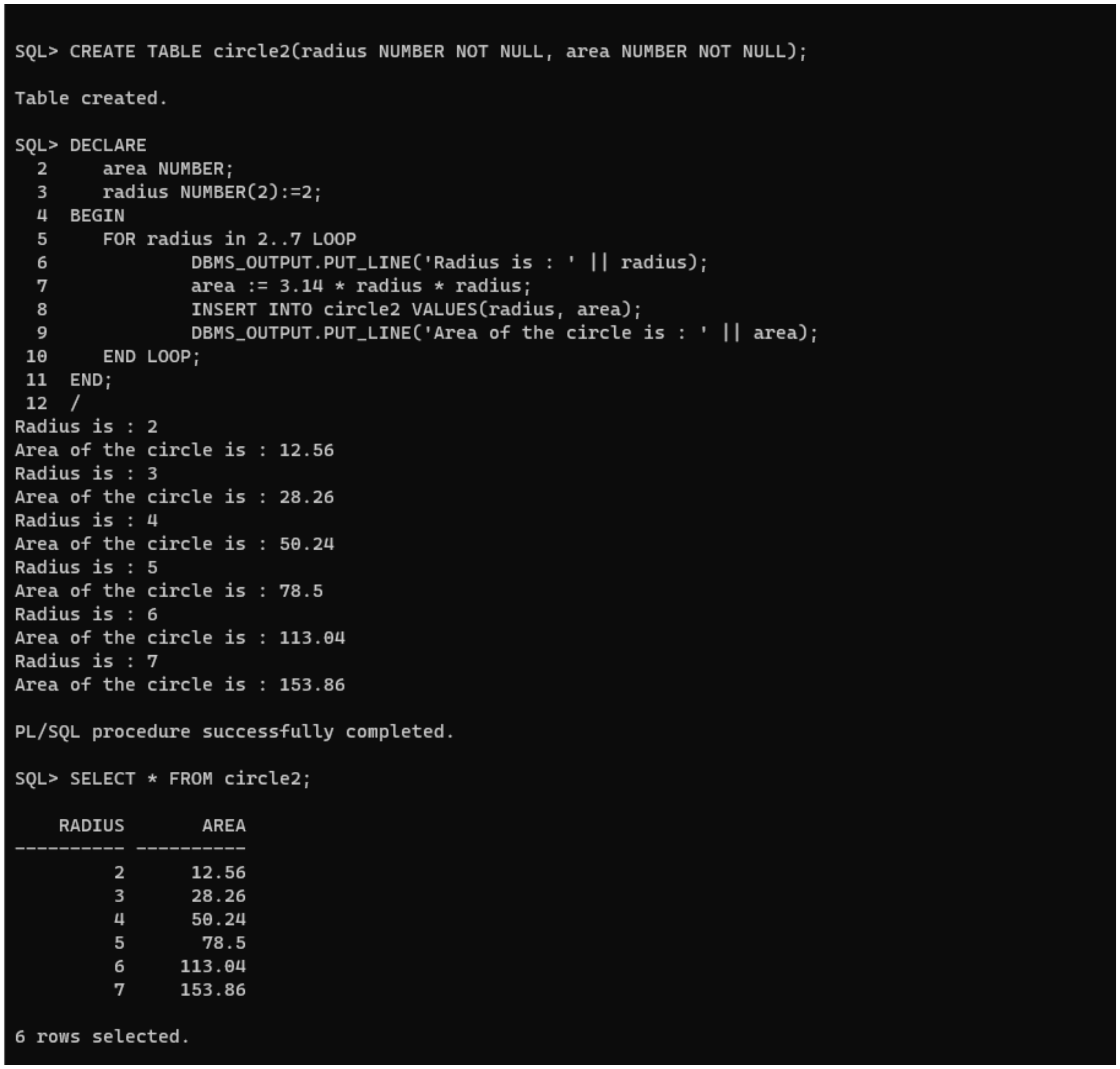


```
DECLARE
   area NUMBER;
   radius NUMBER(2):=2;
BEGIN
   FOR radius in 2..7 LOOP
           DBMS_OUTPUT.PUT_LINE('Radius is : ' || radius);
           area := 3.14 * radius * radius;
           DBMS_OUTPUT.PUT_LINE('Area of the circle is : ' || area);
   END LOOP;
END;
SQL> /
Radius is : 2
Area of the circle is : 12.56
Radius is : 3
Area of the circle is : 28.26
Radius is : 4
Area of the circle is : 50.24
Radius is : 5
Area of the circle is : 78.5
Radius is : 6
Area of the circle is : 113.04
Radius is : 7
Area of the circle is : 153.86

PL/SQL procedure successfully completed.
```

Practical 5

1. Create a table to insert the values of radius and area and use a for loop to print the area from 2 to 7 and insert the data into the table created

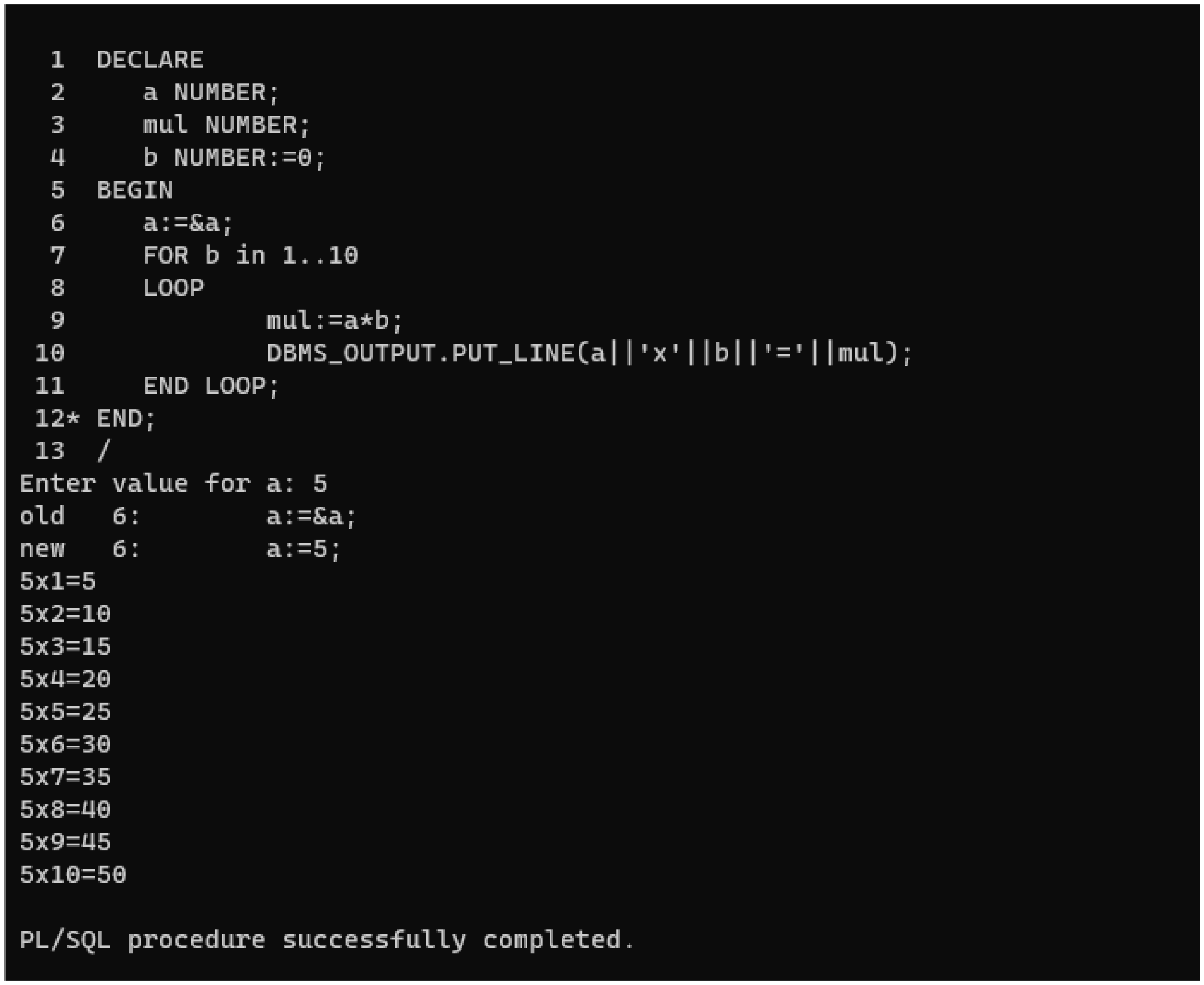
```
CREATE TABLE circle2(radius NUMBER NOT NULL, area NUMBER NOT NULL);

DECLARE
	area NUMBER;
	radius NUMBER(2):=2;
BEGIN
	FOR radius in 2..7 LOOP
		DBMS_OUTPUT.PUT_LINE('Radius is : ' || radius);
		area := 3.14 * radius * radius;
		INSERT INTO circle2 VALUES(radius, area);
		DBMS_OUTPUT.PUT_LINE('Area of the circle is : ' || area);
	END LOOP;
END;
/

SELECT * FROM circle2;
```



```
SQL> CREATE TABLE circle2(radius NUMBER NOT NULL, area NUMBER NOT NULL);

Table created.

SQL> DECLARE
        area NUMBER;
        radius NUMBER(2):=2;
     BEGIN
        FOR radius in 2..7 LOOP
                DBMS_OUTPUT.PUT_LINE('Radius is : ' || radius);
                area := 3.14 * radius * radius;
                INSERT INTO circle2 VALUES(radius, area);
                DBMS_OUTPUT.PUT_LINE('Area of the circle is : ' || area);
        END LOOP;
     END;
     /
Radius is : 2
Area of the circle is : 12.56
Radius is : 3
Area of the circle is : 28.26
Radius is : 4
Area of the circle is : 50.24
Radius is : 5
Area of the circle is : 78.5
Radius is : 6
Area of the circle is : 113.04
Radius is : 7
Area of the circle is : 153.86

PL/SQL procedure successfully completed.

SQL> SELECT * FROM circle2;

    RADIUS       AREA
---------- ----------
         2      12.56
         3      28.26
         4      50.24
         5       78.5
         6     113.04
         7     153.86

6 rows selected.
```

2. Write a PL/SQL code to calculate the table of a number

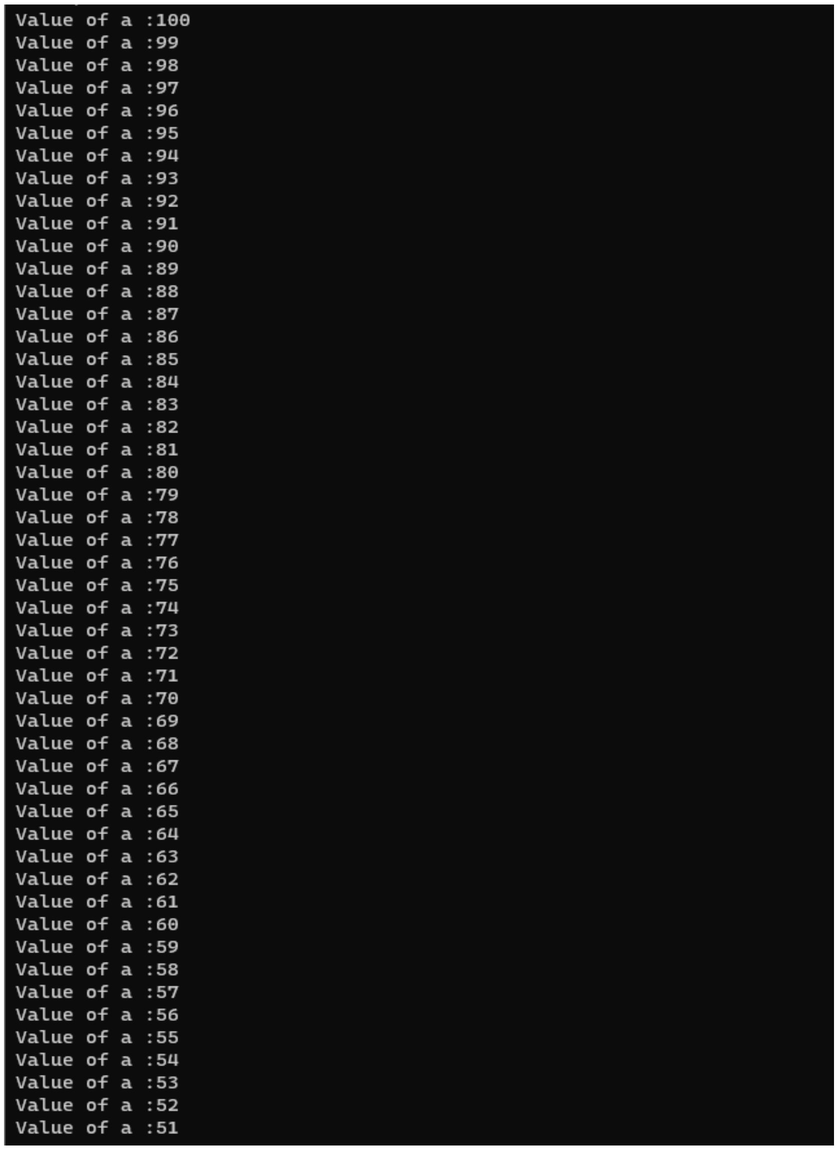
```
DECLARE
	a NUMBER;
	mul NUMBER;
	b NUMBER:=0;
BEGIN
	a:=&a;
	FOR b in 1..10
	LOOP
		mul:=a*b;
		DBMS_OUTPUT.PUT_LINE(a||’x’||b||’=’||mul);
	END LOOP;
END;
/
```



```
DECLARE
   a NUMBER;
   mul NUMBER;
   b NUMBER:=0;
BEGIN
   a:=&a;
   FOR b in 1..10
   LOOP
           mul:=a*b;
           DBMS_OUTPUT.PUT_LINE(a||'x'||b||'='||mul);
   END LOOP;
END;
/
Enter value for a: 5
old   6:        a:=&a;
new   6:        a:=5;
5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50

PL/SQL procedure successfully completed.
```

3. Write a PL/SQL code to display the number in reverse order from 51 to 100 using for loop

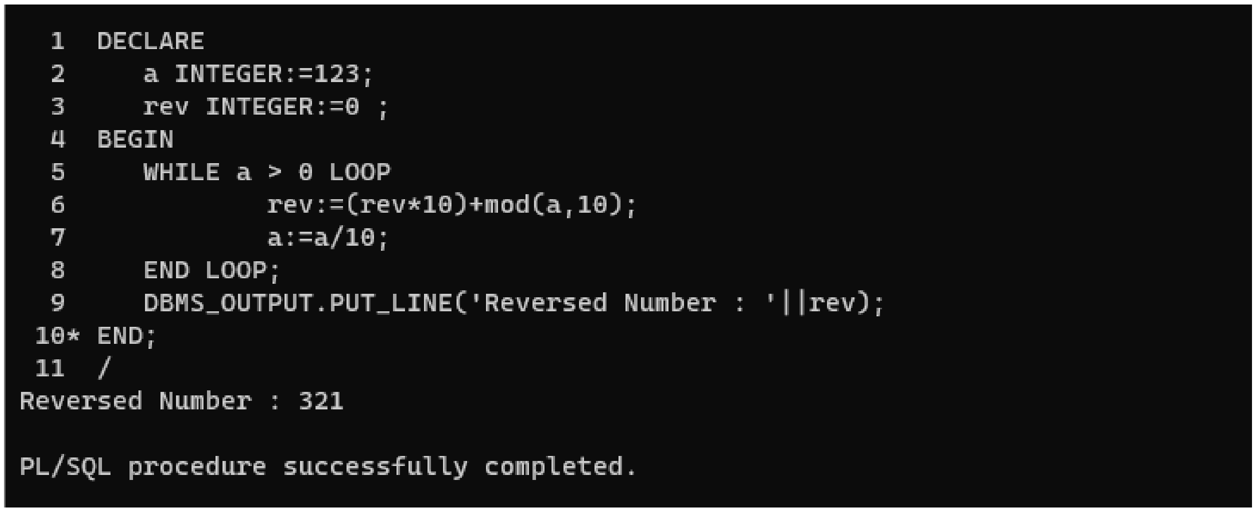
```
DECLARE
        a NUMBER;
BEGIN
        FOR a in REVERSE 51..100
        LOOP
                DBMS_OUTPUT.PUT_LINE('Value of a :'||a);
        END LOOP;
END;
/
```



```
Value of a :100
Value of a :99
Value of a :98
Value of a :97
Value of a :96
Value of a :95
Value of a :94
Value of a :93
Value of a :92
Value of a :91
Value of a :90
Value of a :89
Value of a :88
Value of a :87
Value of a :86
Value of a :85
Value of a :84
Value of a :83
Value of a :82
Value of a :81
Value of a :80
Value of a :79
Value of a :78
Value of a :77
Value of a :76
Value of a :75
Value of a :74
Value of a :73
Value of a :72
Value of a :71
Value of a :70
Value of a :69
Value of a :68
Value of a :67
Value of a :66
Value of a :65
Value of a :64
Value of a :63
Value of a :62
Value of a :61
Value of a :60
Value of a :59
Value of a :58
Value of a :57
Value of a :56
Value of a :55
Value of a :54
Value of a :53
Value of a :52
Value of a :51
```

4. Write a PL/SQL code to display the reverse of a number

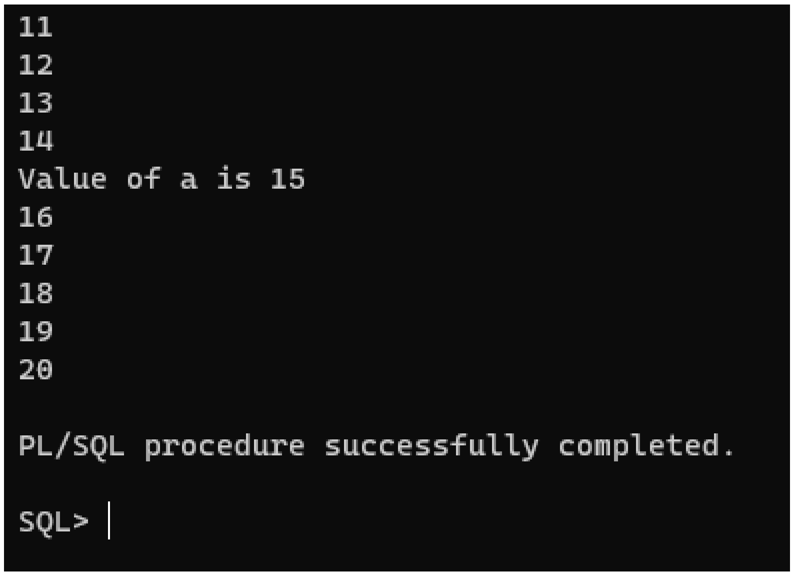
```
DECLARE
        a INTEGER:=123;
        rev INTEGER:=0 ;

BEGIN
        WHILE a > 0 LOOP
                rev:=(rev*10)+mod(a,10);
                a:=a/10;
        END LOOP;
        DBMS_OUTPUT.PUT_LINE('Reversed Number : '||rev);
END;
/
```



```
DECLARE
   a INTEGER:=123;
   rev INTEGER:=0 ;
BEGIN
   WHILE a > 0 LOOP
           rev:=(rev*10)+mod(a,10);
           a:=a/10;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Reversed Number : '||rev);
END;
/
Reversed Number : 321

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL code to display numbers from 11 to 20 but when the number is 15 it mentions the 'value of a is 15' using goto statement.

```
declare
  a number:=11;
  begin
  <<loopstart>>
  while a<=20
  loop
  if a=15 then
  dbms_output.put_line('Value of a is 15');
  a:=a+1;
  goto loopstart;
  end if;
  dbms_output.put_line(a);
  a:=a+1;
  end loop;
  end;
  /
```

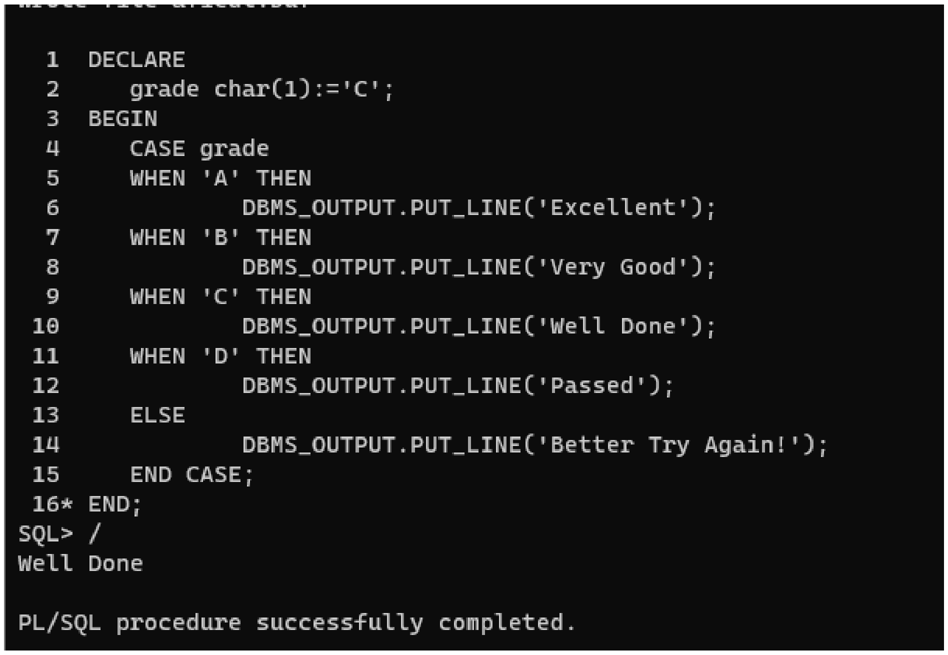


```
11
12
13
14
Value of a is 15
16
17
18
19
20

PL/SQL procedure successfully completed.

SQL>
```

Practical 6

1. Write a PL/SQL Code to display message according to the grade
If A, then display Excellent
If B, then display Very Good
If C, then display Well Done
If D, then display Passed
If E, then display Better try again
Using case statement

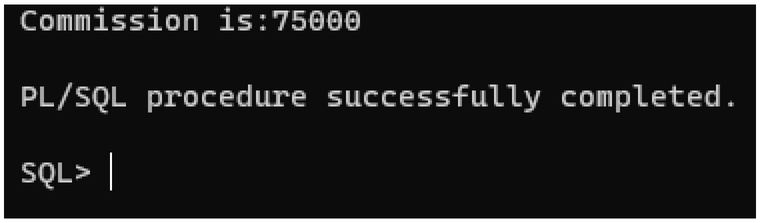
```
DECLARE
        grade char(1):='C';
BEGIN
        CASE grade
        WHEN 'A' THEN
                DBMS_OUTPUT.PUT_LINE('Excellent');
        WHEN 'B' THEN
                DBMS_OUTPUT.PUT_LINE('Very Good');
        WHEN 'C' THEN
                DBMS_OUTPUT.PUT_LINE('Well Done');
        WHEN 'D' THEN
                DBMS_OUTPUT.PUT_LINE('Passed');
        ELSE
                DBMS_OUTPUT.PUT_LINE('Better Try Again!');
        END CASE;
END;
/
```



```
DECLARE
   grade char(1):='C';
BEGIN
   CASE grade
   WHEN 'A' THEN
           DBMS_OUTPUT.PUT_LINE('Excellent');
   WHEN 'B' THEN
           DBMS_OUTPUT.PUT_LINE('Very Good');
   WHEN 'C' THEN
           DBMS_OUTPUT.PUT_LINE('Well Done');
   WHEN 'D' THEN
           DBMS_OUTPUT.PUT_LINE('Passed');
   ELSE
           DBMS_OUTPUT.PUT_LINE('Better Try Again!');
   END CASE;
END;
SQL> /
Well Done

PL/SQL procedure successfully completed.
```

2. Write a PL/SQL code if sales is greater than 200000 then commission is 70% if sales is greater than 100000 but less than 200000 then commission is 50% if sales is greater than 50000 but less than 100000 then commission is 30% else 10% commission to the employees using case statement.

```
declare
    sales number:=150000;
    commission number;
    begin
    Case
    when sales>200000 then
    commission:=sales*0.7;
    dbms_output.put_line('Commission is:'||commission);
    when sales>100000 AND sales<200000 then
   commission:=sales*0.5;
   dbms_output.put_line('Commission is:'||commission);
   when sales>50000 AND sales<100000 then
   commission:=sales*0.3;
   dbms_output.put_line('Commission is:'||commission);
   else
   commission:=sales*0.1;
   dbms_output.put_line('Commission is:'||commission);
   end case;
   end;
   /
```

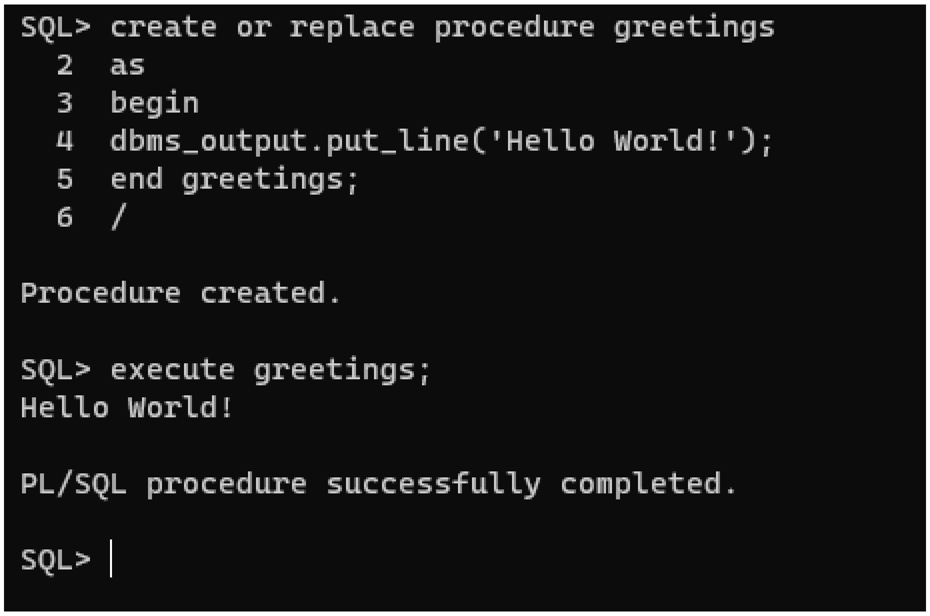


```
Commission is:75000

PL/SQL procedure successfully completed.

SQL>
```

Practical 7

1. Write a program to create a procedure for displaying ‘ HELLO WORLD! ’ message.

```
create or replace procedure greetings
 as
 begin
 dbms_output.put_line('Hello World!');
 end greetings;
 /

SQL> execute greetings;
Hello World!
```

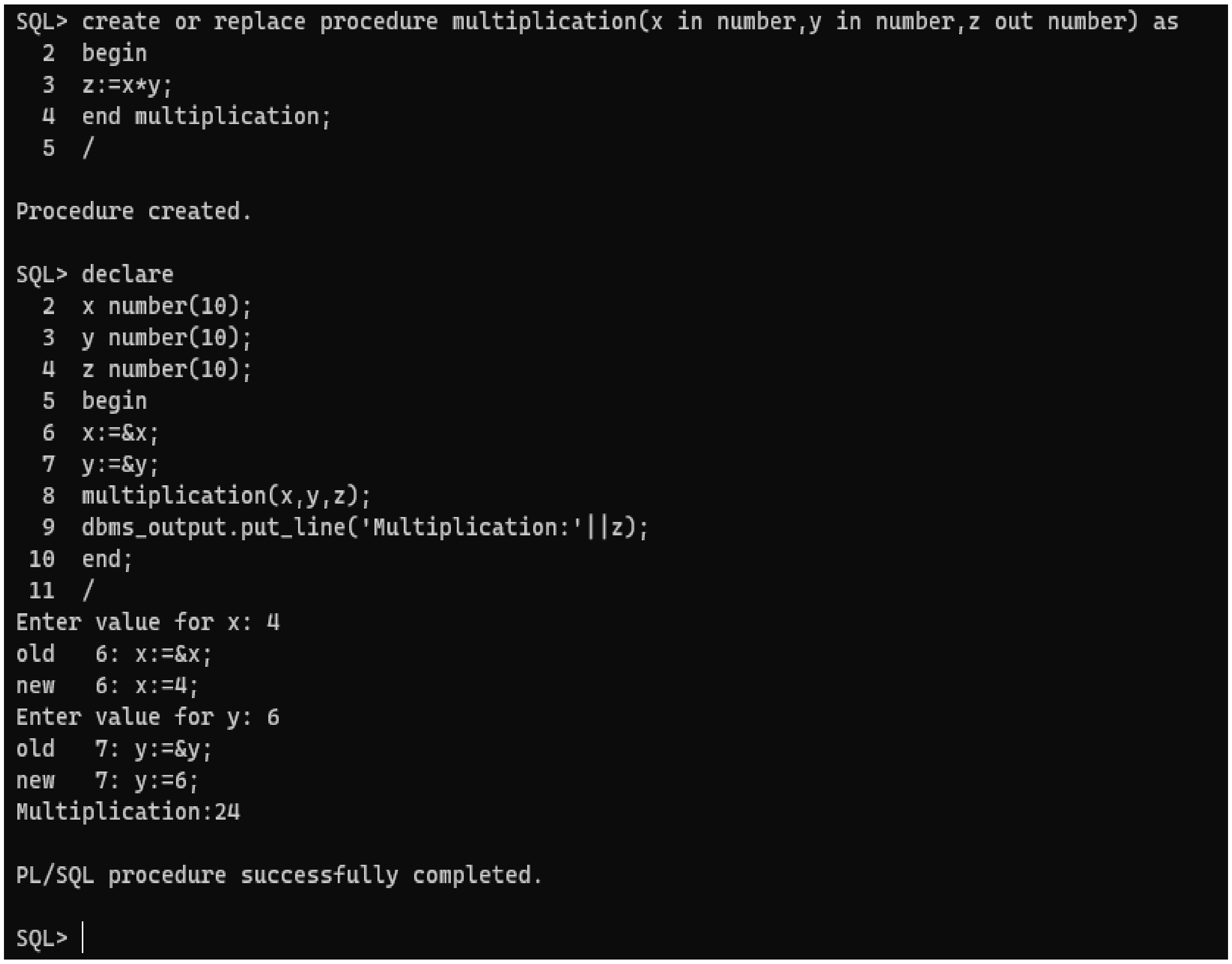


```
SQL> create or replace procedure greetings
  2  as
  3  begin
  4  dbms_output.put_line('Hello World!');
  5  end greetings;
  6  /

Procedure created.

SQL> execute greetings;
Hello World!

PL/SQL procedure successfully completed.

SQL>
```

2. Write a program to create a procedure for multiplication of two numbers.

```
create or replace procedure multiplication(x in number,y in number,z out number) as
begin
z:=x*y;
end multiplication;
/

declare
x number(10);
y number(10);
z number(10);
begin
x:=&x;
y:=&y;
multiplication(x,y,z);
dbms_output.put_line('Multiplication:'||z);
end;
/
```

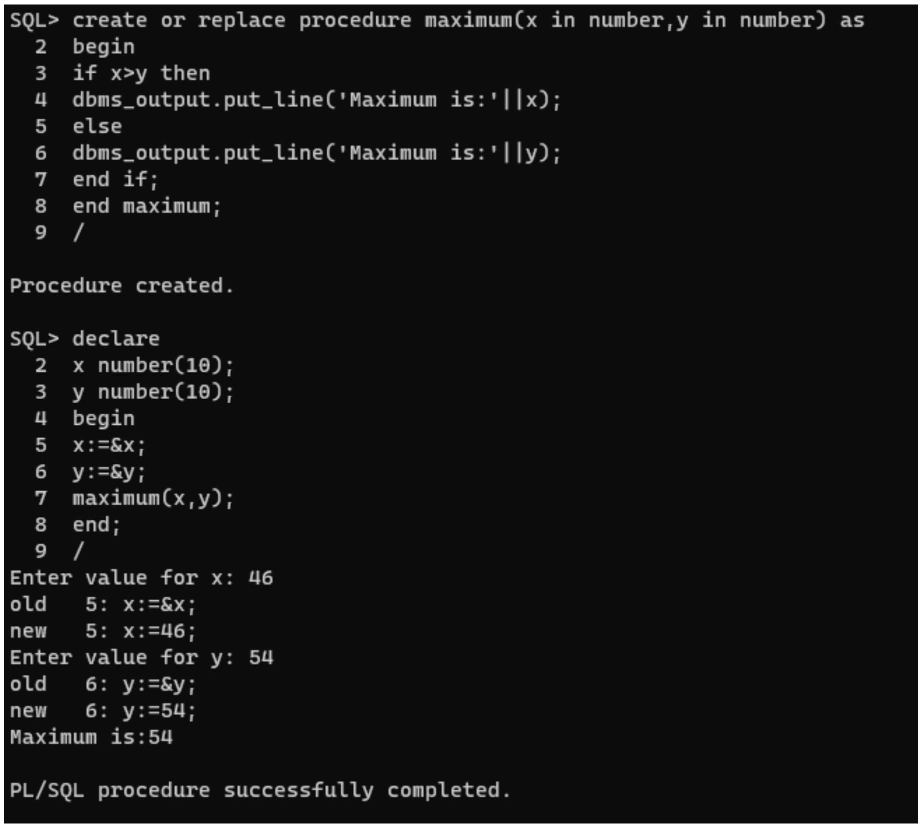
OUTPUT:-



```
SQL> create or replace procedure multiplication(x in number,y in number,z out number) as
  2  begin
  3  z:=x*y;
  4  end multiplication;
  5  /

Procedure created.

SQL> declare
  2  x number(10);
  3  y number(10);
  4  z number(10);
  5  begin
  6  x:=&x;
  7  y:=&y;
  8  multiplication(x,y,z);
  9  dbms_output.put_line('Multiplication:'||z);
 10  end;
 11  /
Enter value for x: 4
old   6: x:=&x;
new   6: x:=4;
Enter value for y: 6
old   7: y:=&y;
new   7: y:=6;
Multiplication:24

PL/SQL procedure successfully completed.

SQL>
```

3. Write a program to create a procedure for displaying a maximum of two numbers.

```
create or replace procedure maximum(x in number,y in number) as
begin
if x>y then
dbms_output.put_line('Maximum is:'||x);
else
dbms_output.put_line('Maximum is:'||y);
end if;
end maximum;
/

declare
x number(10);
y number(10);
begin
x:=&x;
y:=&y;
maximum(x,y);
end;
/
```

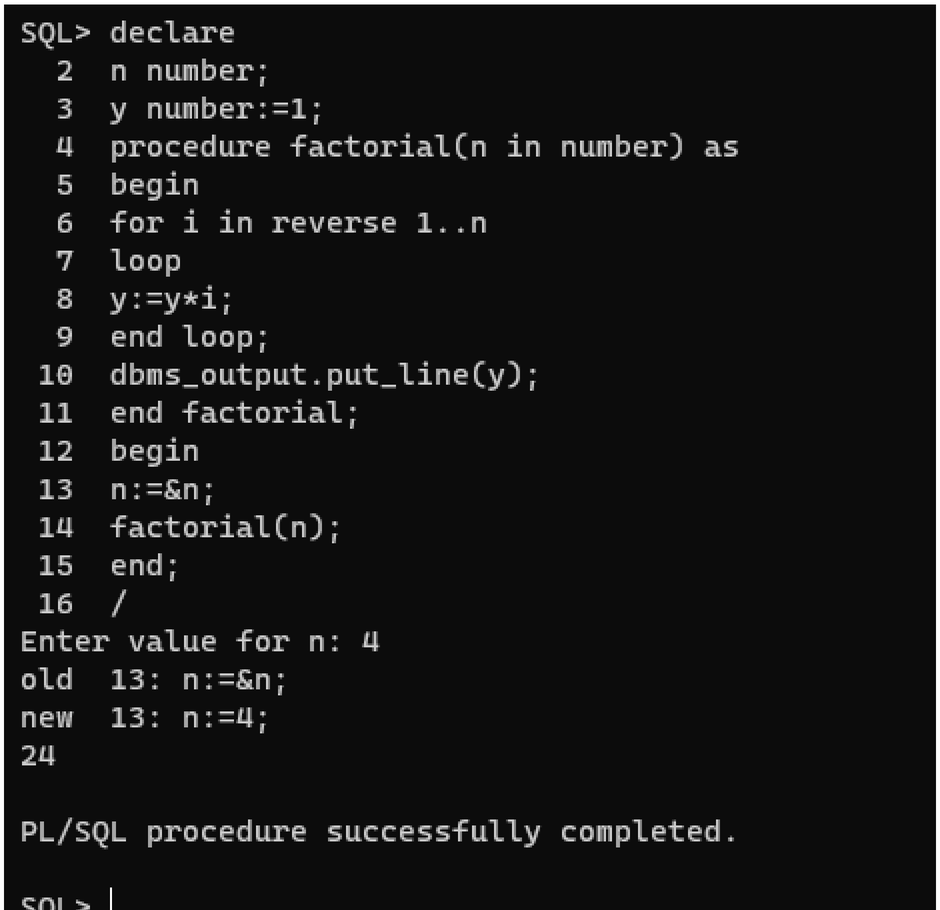


```
SQL> create or replace procedure maximum(x in number,y in number) as
  2  begin
  3  if x>y then
  4  dbms_output.put_line('Maximum is:'||x);
  5  else
  6  dbms_output.put_line('Maximum is:'||y);
  7  end if;
  8  end maximum;
  9  /

Procedure created.

SQL> declare
  2  x number(10);
  3  y number(10);
  4  begin
  5  x:=&x;
  6  y:=&y;
  7  maximum(x,y);
  8  end;
  9  /
Enter value for x: 46
old   5: x:=&x;
new   5: x:=46;
Enter value for y: 54
old   6: y:=&y;
new   6: y:=54;
Maximum is:54

PL/SQL procedure successfully completed.
```

4. Write a program to create procedure inside PL/SQL code to find a factorial of a number.

```
declare
n number;
y number:=1;
procedure factorial(n in number) as
begin
for i in reverse 1..n
loop
y:=y*i;
end loop;
dbms_output.put_line(y);
end factorial;
begin
n:=&n;
factorial(n);
end;
/
```

OUTPUT:-

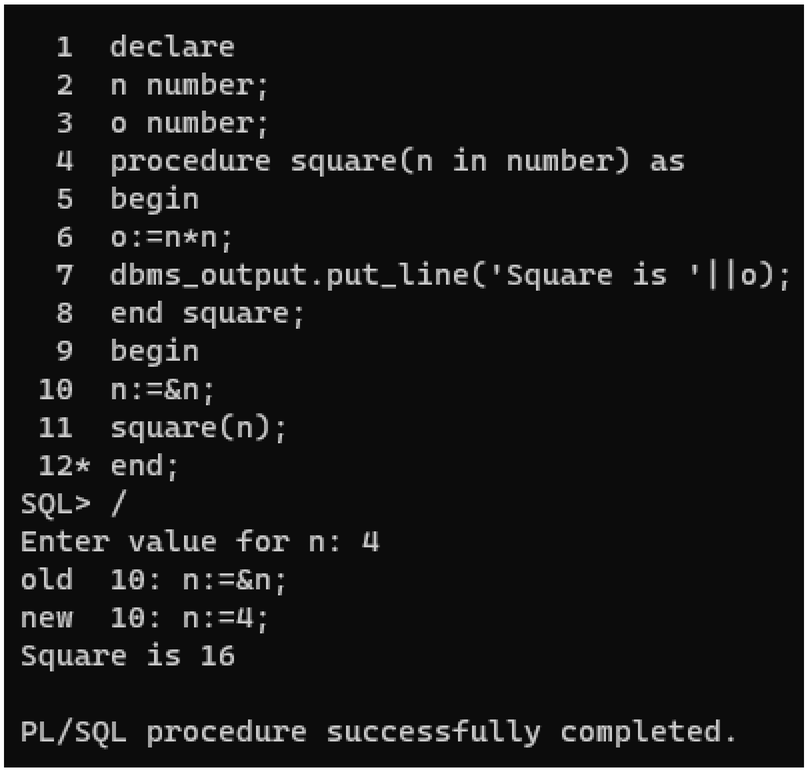


```
SQL> declare
  2  n number;
  3  y number:=1;
  4  procedure factorial(n in number) as
  5  begin
  6  for i in reverse 1..n
  7  loop
  8  y:=y*i;
  9  end loop;
 10  dbms_output.put_line(y);
 11  end factorial;
 12  begin
 13  n:=&n;
 14  factorial(n);
 15  end;
 16  /
Enter value for n: 4
old  13: n:=&n;
new  13: n:=4;
24

PL/SQL procedure successfully completed.

SQL>
```

5. Write a program to create a procedure inside PL/SQL code to find a square of a number.

```
declare
n number;
o number;
procedure square(n in number) as
begin
o:=n*n;
dbms_output.put_line('Square is '||o);
end square;
begin
n:=&n;
square(n);
end;
/
```

Output:

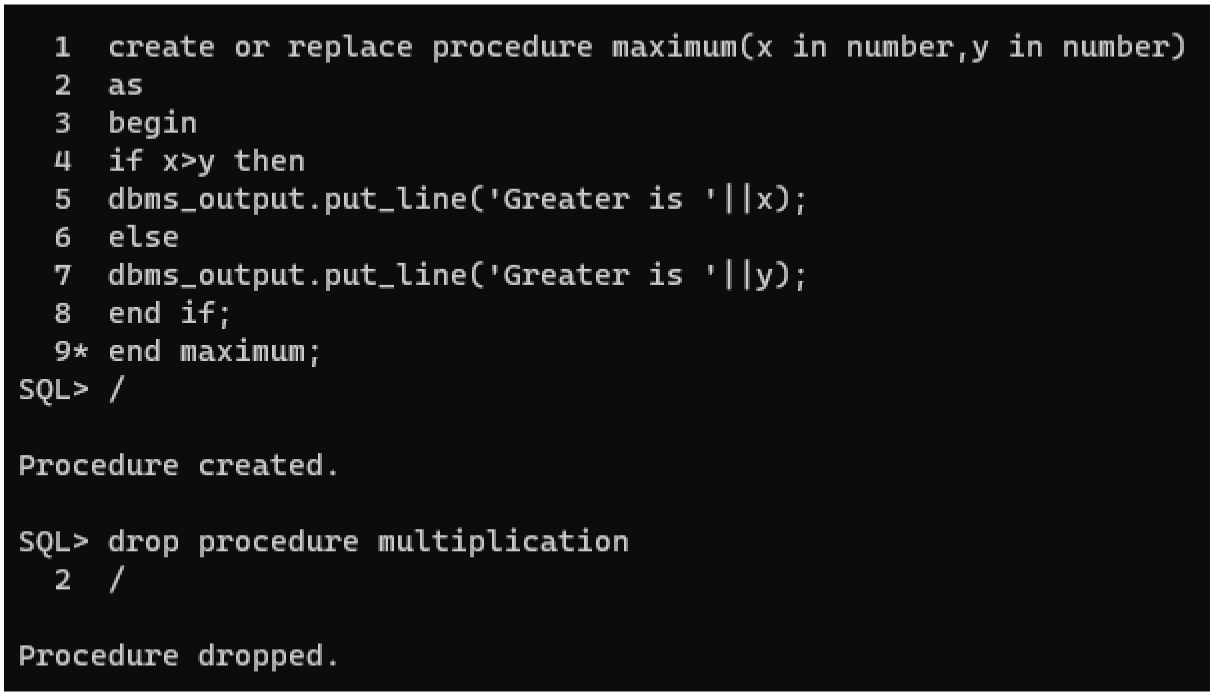


```
declare
n number;
o number;
procedure square(n in number) as
begin
o:=n*n;
dbms_output.put_line('Square is '||o);
end square;
begin
n:=&n;
square(n);
end;
SQL> /
Enter value for n: 4
old  10: n:=&n;
new  10: n:=4;
Square is 16

PL/SQL procedure successfully completed.
```

6. Write a program to drop procedure inside PL/SQL.

```
create or replace procedure maximum(x in number,y in number)
as
begin
if x>y then
dbms_output.put_line('Greater is '||x);
else
dbms_output.put_line('Greater is '||y);
end if;
end maximum;
/

drop procedure multiplication
/
```

Output:

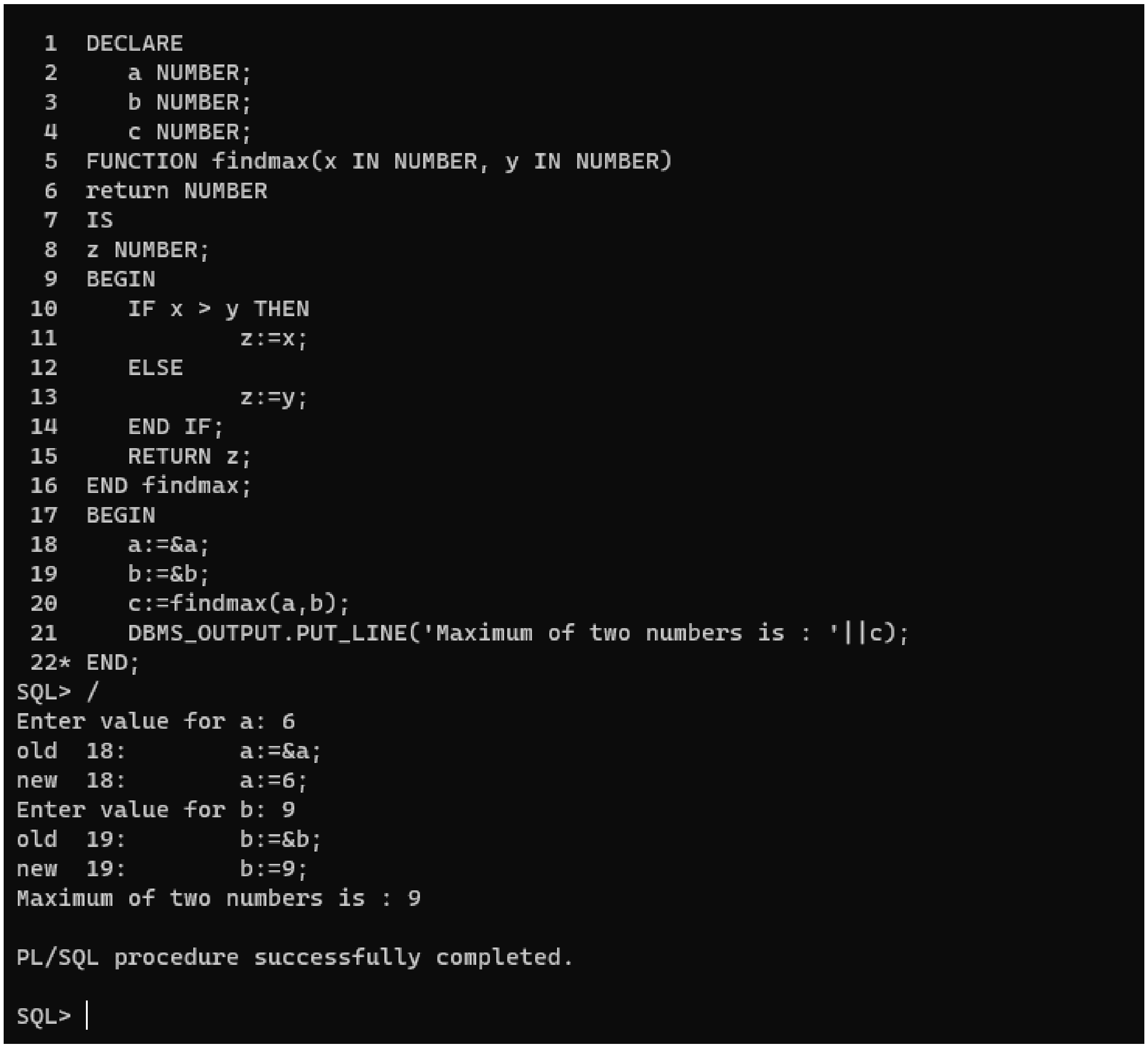


```
create or replace procedure maximum(x in number,y in number)
as
begin
if x>y then
dbms_output.put_line('Greater is '||x);
else
dbms_output.put_line('Greater is '||y);
end if;
end maximum;
SQL> /

Procedure created.

SQL> drop procedure multiplication
/

Procedure dropped.
```

Practical 8

1. Write a PL/SQL code to create a function to find the maximum of two numbers

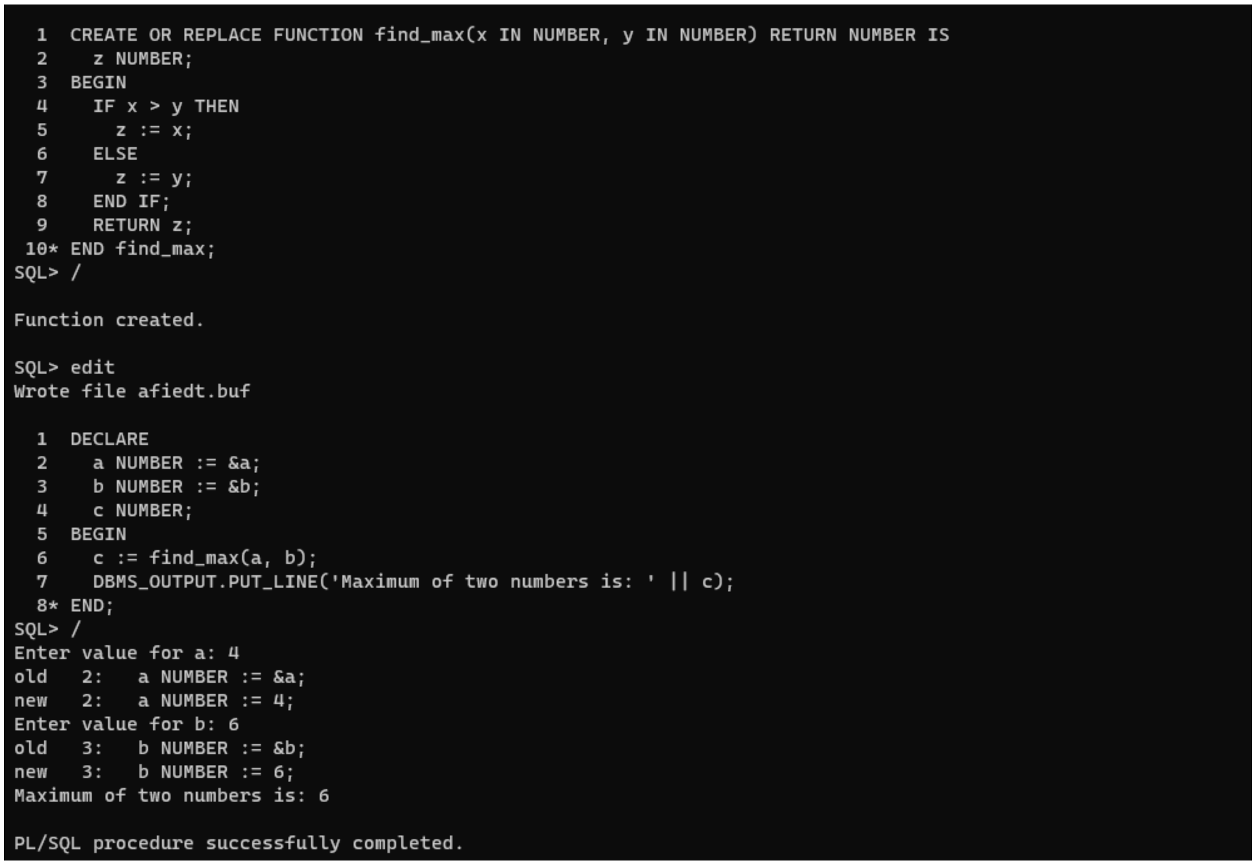
```
DECLARE
        a NUMBER;
        b NUMBER;
        c NUMBER;
FUNCTION findmax(x IN NUMBER, y IN NUMBER)
return NUMBER
IS
z NUMBER;
BEGIN
        IF x > y THEN
                z:=x;
        ELSE
                z:=y;
        END IF;
        RETURN z;
END findmax;
BEGIN
        a:=&a;
        b:=&b;
        c:=findmax(a,b);
        DBMS_OUTPUT.PUT_LINE('Maximum of two numbers is : '||c);
END;
/
```



```
  1  DECLARE
  2     a NUMBER;
  3     b NUMBER;
  4     c NUMBER;
  5  FUNCTION findmax(x IN NUMBER, y IN NUMBER)
  6  return NUMBER
  7  IS
  8  z NUMBER;
  9  BEGIN
 10     IF x > y THEN
 11             z:=x;
 12     ELSE
 13             z:=y;
 14     END IF;
 15     RETURN z;
 16  END findmax;
 17  BEGIN
 18     a:=&a;
 19     b:=&b;
 20     c:=findmax(a,b);
 21     DBMS_OUTPUT.PUT_LINE('Maximum of two numbers is : '||c);
 22* END;
SQL> /
Enter value for a: 6
old  18:        a:=&a;
new  18:        a:=6;
Enter value for b: 9
old  19:        b:=&b;
new  19:        b:=9;
Maximum of two numbers is : 9

PL/SQL procedure successfully completed.

SQL> |
```

2. Write a PL/SQL code to create a function to find the maximum of two numbers using standalone function

```
CREATE OR REPLACE FUNCTION find_max(x IN NUMBER, y IN NUMBER)
RETURN NUMBER IS
  z NUMBER;
BEGIN
  IF x > y THEN
    z := x;
  ELSE
    z := y;
  END IF;
  RETURN z;
END find_max;
/

DECLARE
  a NUMBER := &a;
  b NUMBER := &b;
  c NUMBER;
BEGIN
  c := find_max(a, b);
  DBMS_OUTPUT.PUT_LINE('Maximum of two numbers is: ' || c);
END;
/
```

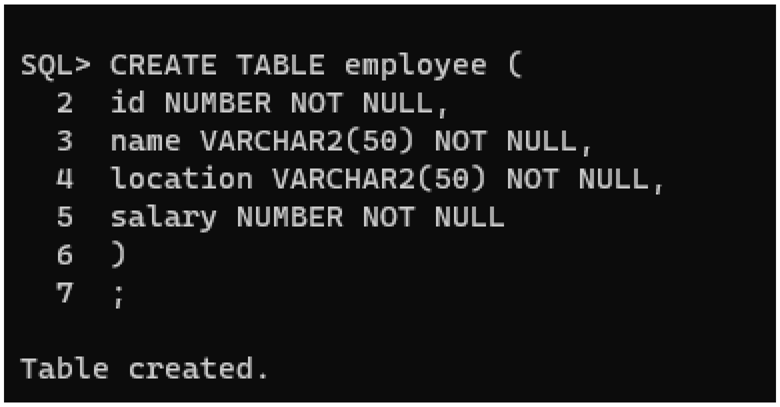


```
  1  CREATE OR REPLACE FUNCTION find_max(x IN NUMBER, y IN NUMBER) RETURN NUMBER IS
  2    z NUMBER;
  3  BEGIN
  4    IF x > y THEN
  5      z := x;
  6    ELSE
  7      z := y;
  8    END IF;
  9    RETURN z;
 10* END find_max;
SQL> /

Function created.

SQL> edit
Wrote file afiedt.buf

  1  DECLARE
  2    a NUMBER := &a;
  3    b NUMBER := &b;
  4    c NUMBER;
  5  BEGIN
  6    c := find_max(a, b);
  7    DBMS_OUTPUT.PUT_LINE('Maximum of two numbers is: ' || c);
  8* END;
SQL> /
Enter value for a: 4
old   2:   a NUMBER := &a;
new   2:   a NUMBER := 4;
Enter value for b: 6
old   3:   b NUMBER := &b;
new   3:   b NUMBER := 6;
Maximum of two numbers is: 6

PL/SQL procedure successfully completed.
```

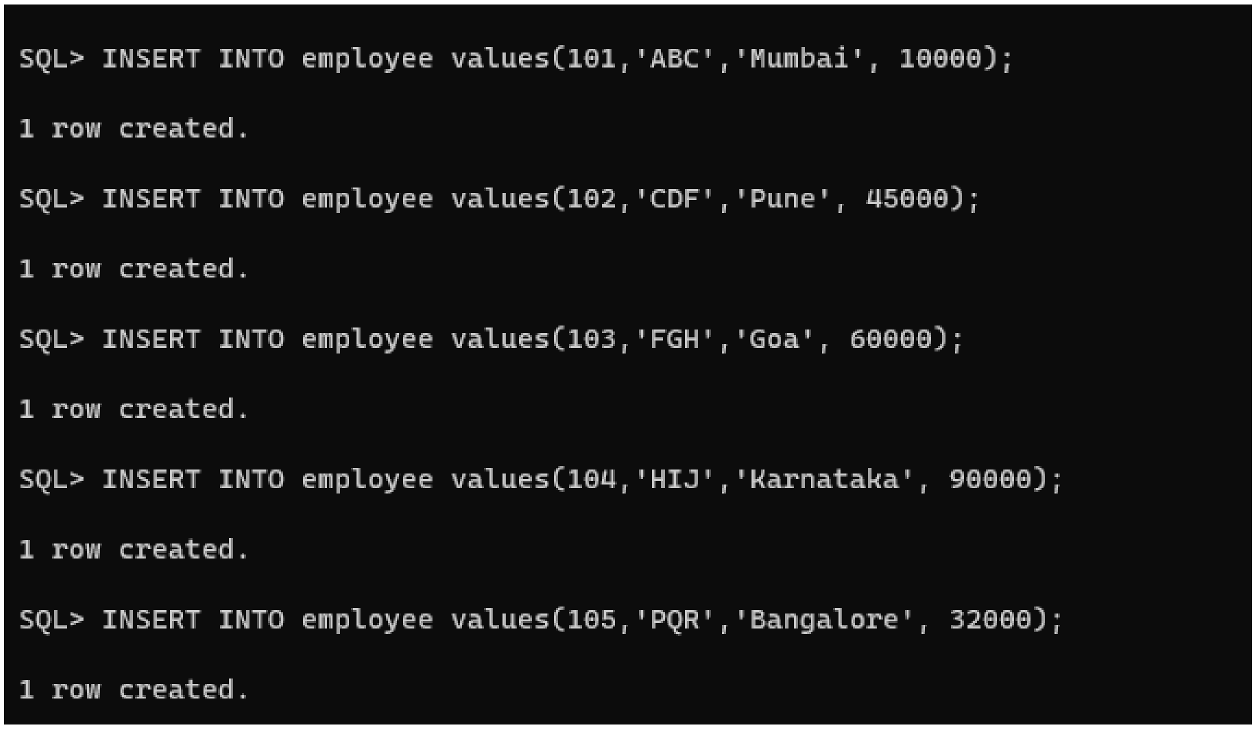
3. Create a function in PL/SQL to calculate the average salary of an employee from the table. Create the table employee and insert 5 records into the table.

Table :

```
CREATE TABLE employee (
   id NUMBER NOT NULL,
   name VARCHAR2(50) NOT NULL,
   location VARCHAR2(50) NOT NULL,
   salary NUMBER NOT NULL
   );
```

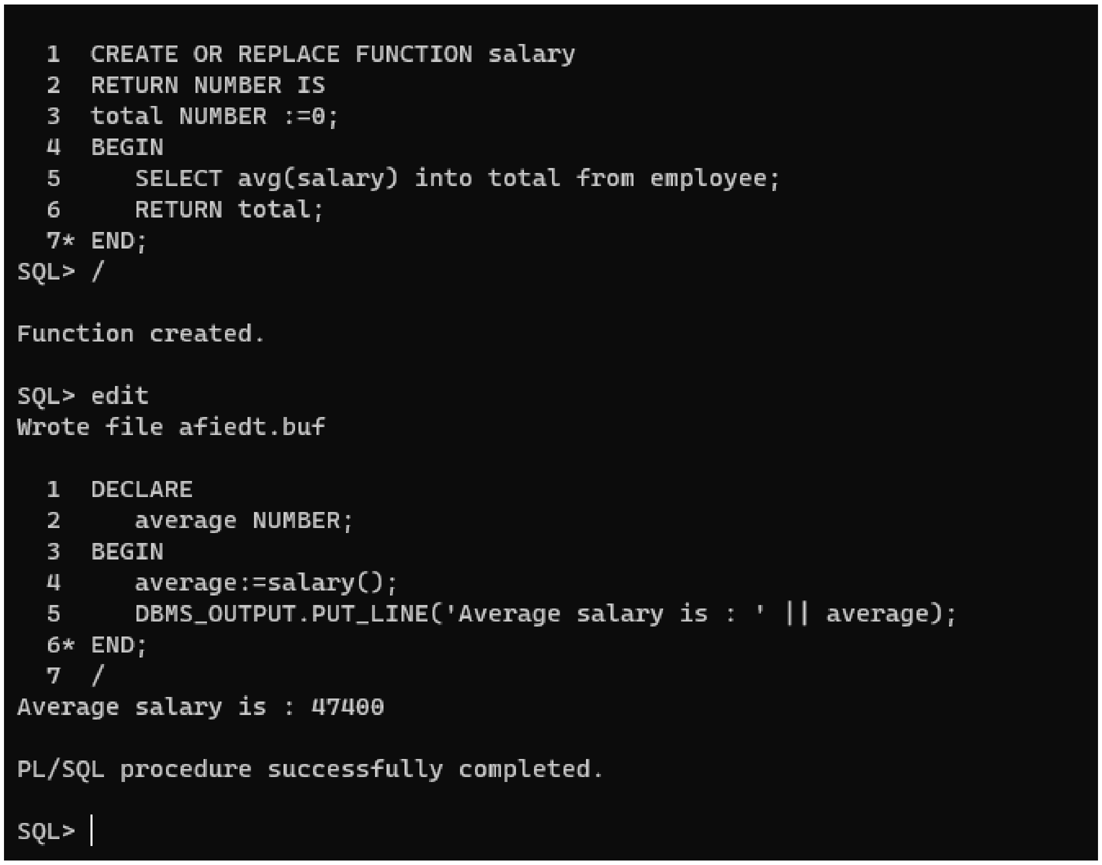


```
SQL> CREATE TABLE employee (
  2  id NUMBER NOT NULL,
  3  name VARCHAR2(50) NOT NULL,
  4  location VARCHAR2(50) NOT NULL,
  5  salary NUMBER NOT NULL
  6  )
  7  ;

Table created.
```


```
SQL> INSERT INTO employee values(101,'ABC','Mumbai', 10000);

1 row created.

SQL> INSERT INTO employee values(102,'CDF','Pune', 45000);

1 row created.

SQL> INSERT INTO employee values(103,'FGH','Goa', 60000);

1 row created.

SQL> INSERT INTO employee values(104,'HIJ','Karnataka', 90000);

1 row created.

SQL> INSERT INTO employee values(105,'PQR','Bangalore', 32000);

1 row created.
```

Code :

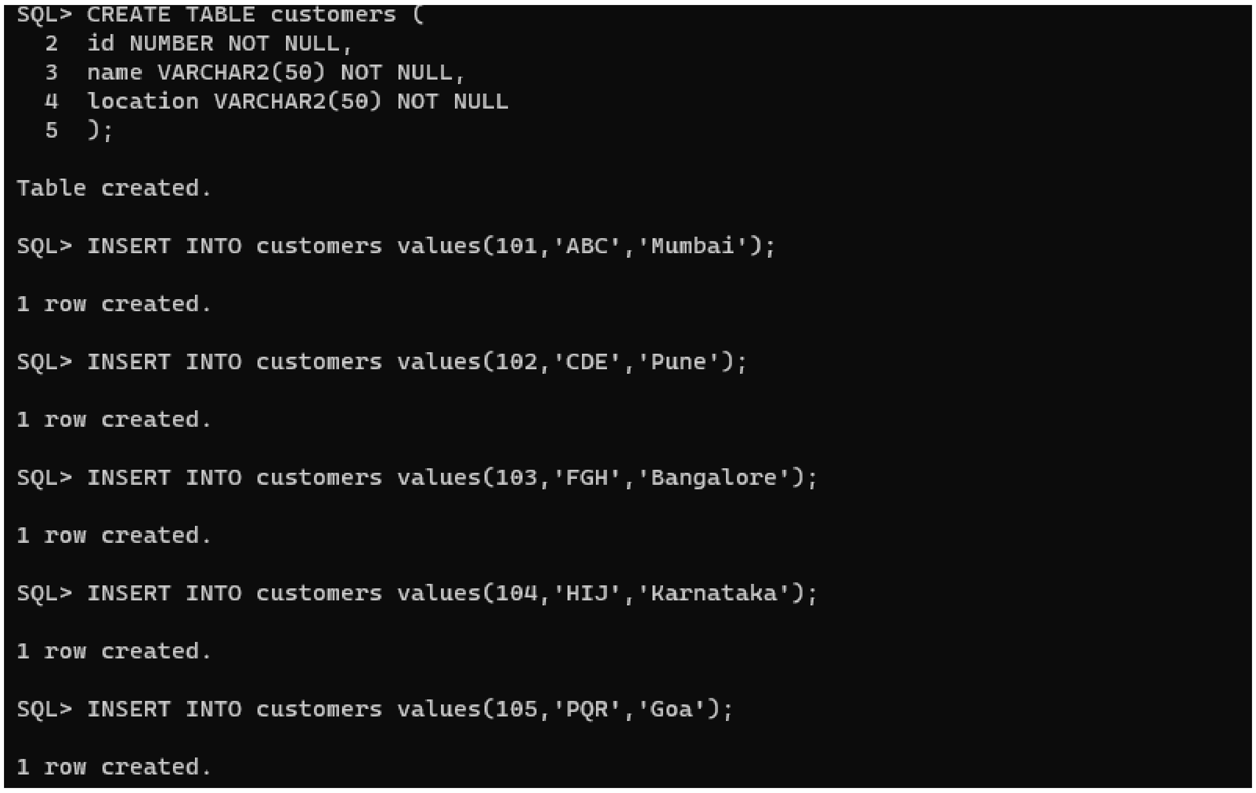
```
CREATE OR REPLACE FUNCTION salary
RETURN NUMBER IS
total NUMBER :=0;
BEGIN
        SELECT avg(salary) into total from employee;
        RETURN total;
END;
/
DECLARE
        average NUMBER;
BEGIN
        average:=salary();
        DBMS_OUTPUT.PUT_LINE('Average salary is : ' || average);
END;
```



```
CREATE OR REPLACE FUNCTION salary
RETURN NUMBER IS
total NUMBER :=0;
BEGIN
   SELECT avg(salary) into total from employee;
   RETURN total;
END;
SQL> /

Function created.

SQL> edit
Wrote file afiedt.buf

DECLARE
   average NUMBER;
BEGIN
   average:=salary();
   DBMS_OUTPUT.PUT_LINE('Average salary is : ' || average);
END;
/
Average salary is : 47400

PL/SQL procedure successfully completed.

SQL>
```

4. **Create a function in PL/SQL to calculate the total number of customers from the table. Create the table customers and insert 5 records into the table.**

Table :

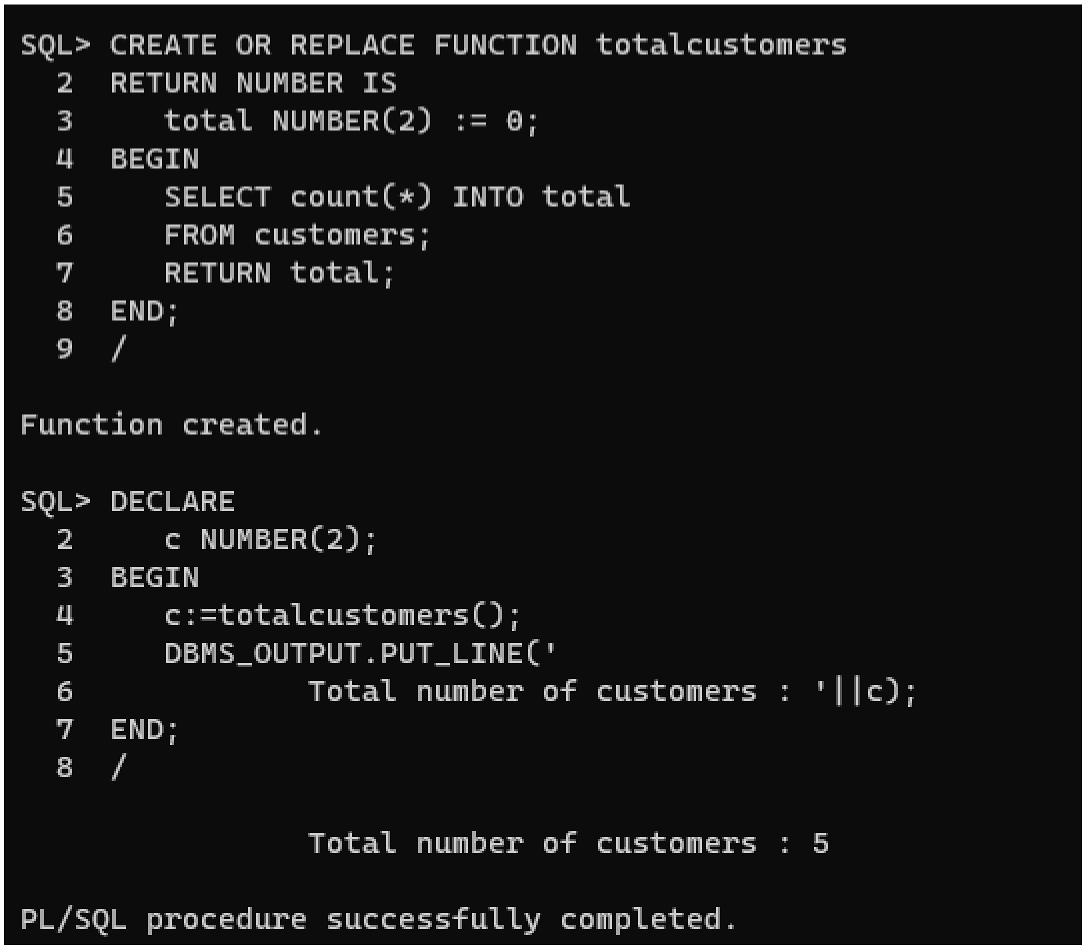


```
SQL> CREATE TABLE customers (
  2  id NUMBER NOT NULL,
  3  name VARCHAR2(50) NOT NULL,
  4  location VARCHAR2(50) NOT NULL
  5  );

Table created.

SQL> INSERT INTO customers values(101,'ABC','Mumbai');

1 row created.

SQL> INSERT INTO customers values(102,'CDE','Pune');

1 row created.

SQL> INSERT INTO customers values(103,'FGH','Bangalore');

1 row created.

SQL> INSERT INTO customers values(104,'HIJ','Karnataka');

1 row created.

SQL> INSERT INTO customers values(105,'PQR','Goa');

1 row created.
```

Code :

```
CREATE OR REPLACE FUNCTION totalcustomers
RETURN NUMBER IS
    total NUMBER(2) := 0;
BEGIN
    SELECT count(*) INTO total
    FROM customers;
   RETURN total;
END;
/


DECLARE
```

```
        c NUMBER(2);
BEGIN
        c:=totalcustomers();
        DBMS_OUTPUT.PUT_LINE('Total number of customers : '||c);
END;
/
```



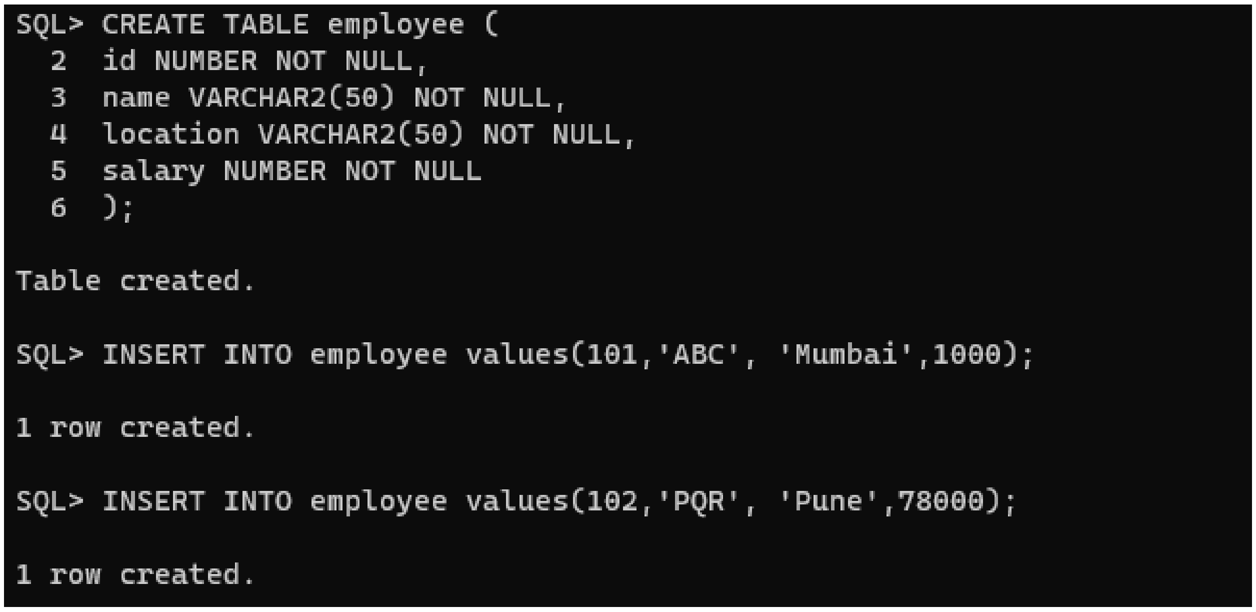
```
SQL> CREATE OR REPLACE FUNCTION totalcustomers
  RETURN NUMBER IS
     total NUMBER(2) := 0;
  BEGIN
     SELECT count(*) INTO total
     FROM customers;
     RETURN total;
  END;
  /

Function created.

SQL> DECLARE
     c NUMBER(2);
  BEGIN
     c:=totalcustomers();
     DBMS_OUTPUT.PUT_LINE('
             Total number of customers : '||c);
  END;
  /

             Total number of customers : 5

PL/SQL procedure successfully completed.
```

Practical 9

1. Create an employee table and insert 2 values into it. Create a trigger that will not allow the user to insert any record into the employee table

Table :

```
 CREATE TABLE employee (
  id NUMBER NOT NULL,
  name VARCHAR2(50) NOT NULL,
  location VARCHAR2(50) NOT NULL,
  salary NUMBER NOT NULL
  );

INSERT INTO employee values(101,'ABC', 'Mumbai',1000);
INSERT INTO employee values(102,'PQR', 'Pune',78000);
```

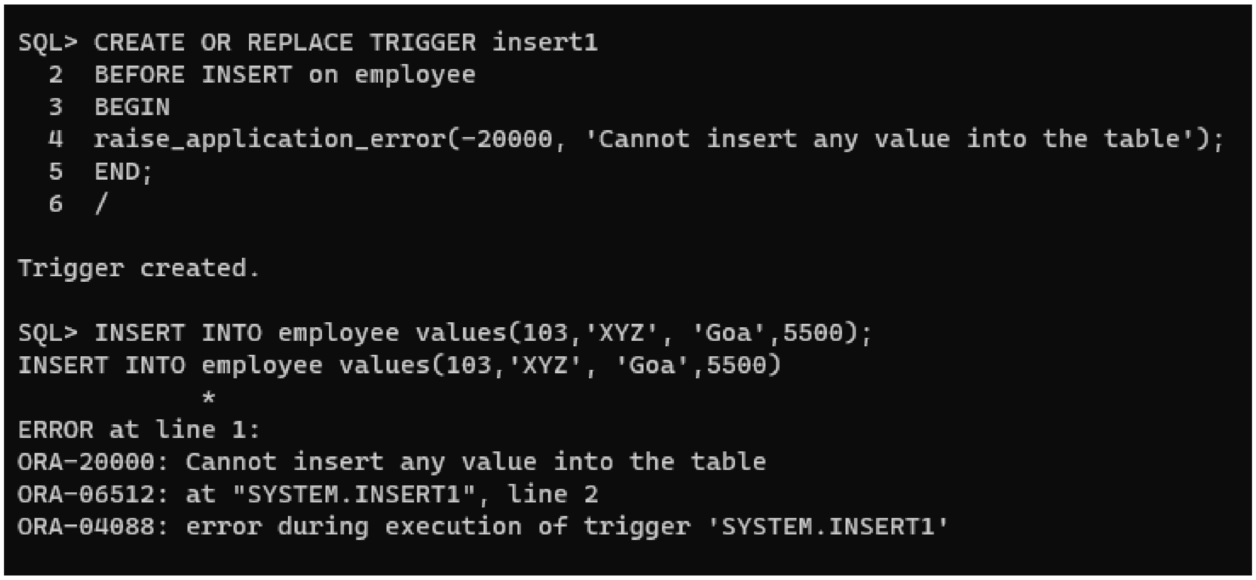


```
SQL> CREATE TABLE employee (
  2  id NUMBER NOT NULL,
  3  name VARCHAR2(50) NOT NULL,
  4  location VARCHAR2(50) NOT NULL,
  5  salary NUMBER NOT NULL
  6  );

Table created.

SQL> INSERT INTO employee values(101,'ABC', 'Mumbai',1000);

1 row created.

SQL> INSERT INTO employee values(102,'PQR', 'Pune',78000);

1 row created.
```

Code :

```
CREATE OR REPLACE TRIGGER insert1
  BEFORE INSERT on employee
  BEGIN
```

```
raise_application_error(-20000, 'Cannot insert any value into the table');
END;
/
```

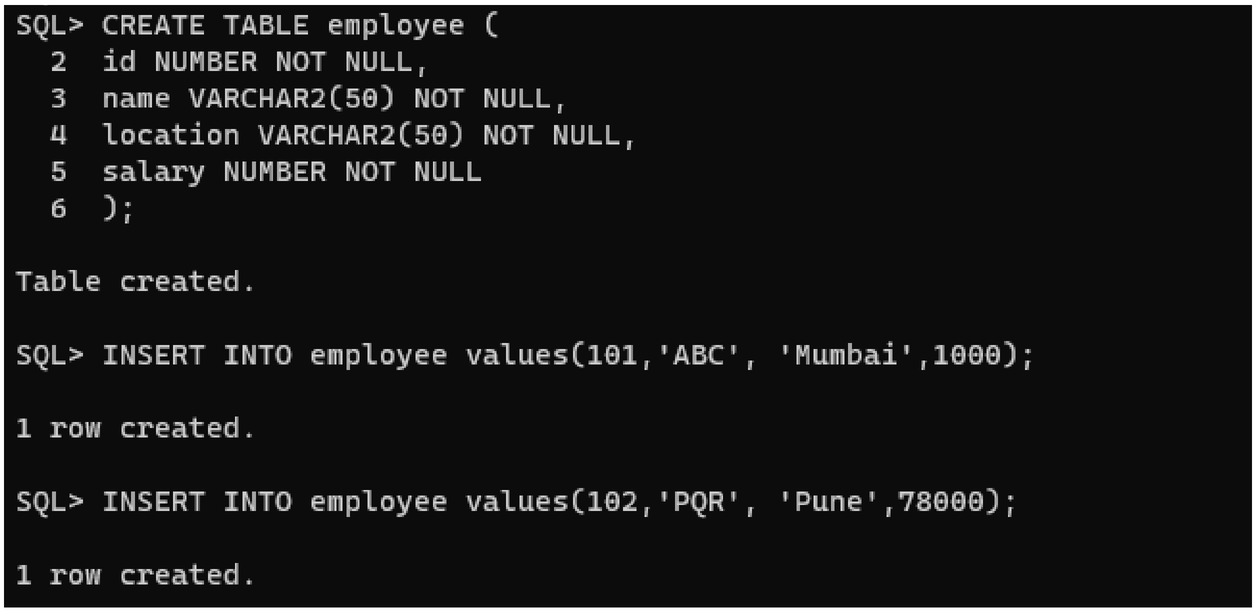


```
SQL> CREATE OR REPLACE TRIGGER insert1
  2  BEFORE INSERT on employee
  3  BEGIN
  4  raise_application_error(-20000, 'Cannot insert any value into the table');
  5  END;
  6  /

Trigger created.

SQL> INSERT INTO employee values(103,'XYZ', 'Goa',5500);
INSERT INTO employee values(103,'XYZ', 'Goa',5500)
            *
ERROR at line 1:
ORA-20000: Cannot insert any value into the table
ORA-06512: at "SYSTEM.INSERT1", line 2
ORA-04088: error during execution of trigger 'SYSTEM.INSERT1'
```

2. Create a trigger which will not allow the user to insert the record whose id is greater than 100 into the employee table

Table :

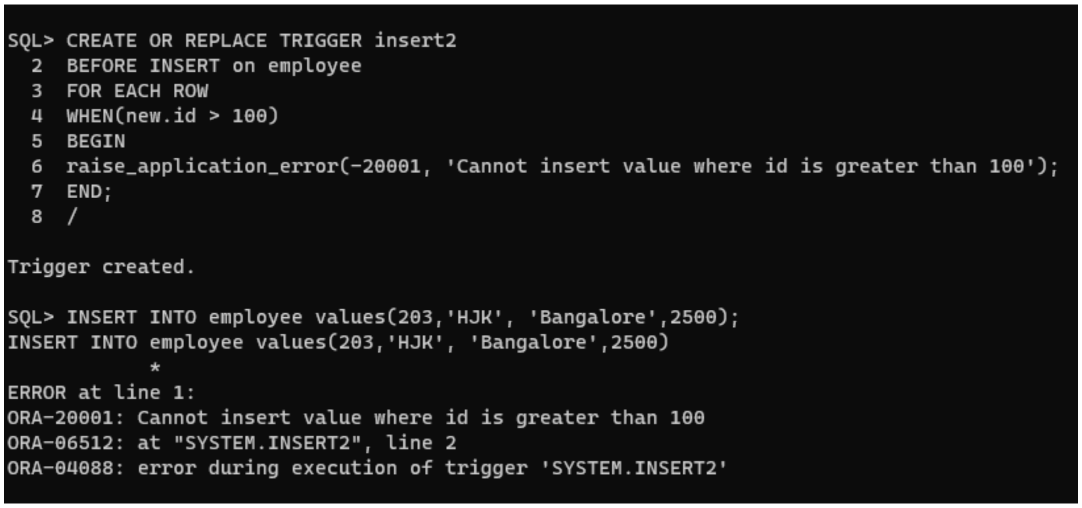
```
 CREATE TABLE employee (
  id NUMBER NOT NULL,
  name VARCHAR2(50) NOT NULL,
  location VARCHAR2(50) NOT NULL,
  salary NUMBER NOT NULL
  );

INSERT INTO employee values(101,'ABC', 'Mumbai',1000);
INSERT INTO employee values(102,'PQR', 'Pune',78000);
```



```
SQL> CREATE TABLE employee (
  2  id NUMBER NOT NULL,
  3  name VARCHAR2(50) NOT NULL,
  4  location VARCHAR2(50) NOT NULL,
  5  salary NUMBER NOT NULL
  6  );

Table created.

SQL> INSERT INTO employee values(101,'ABC', 'Mumbai',1000);

1 row created.

SQL> INSERT INTO employee values(102,'PQR', 'Pune',78000);

1 row created.
```

Code :

```
 CREATE OR REPLACE TRIGGER insert2
    BEFORE INSERT on employee
    FOR EACH ROW
    WHEN(new.id > 100)
    BEGIN
    raise_application_error(-20001, 'Cannot insert value where id is greater
than 100');
    END;
    /
```

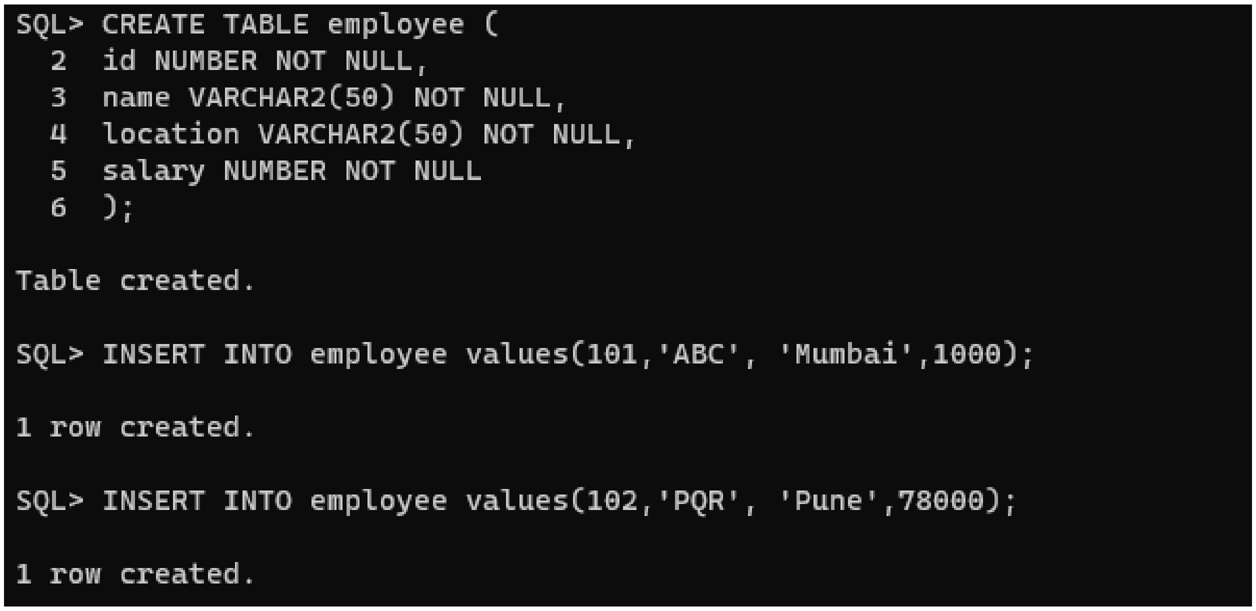


```
SQL> CREATE OR REPLACE TRIGGER insert2
  2  BEFORE INSERT on employee
  3  FOR EACH ROW
  4  WHEN(new.id > 100)
  5  BEGIN
  6  raise_application_error(-20001, 'Cannot insert value where id is greater than 100');
  7  END;
  8  /

Trigger created.

SQL> INSERT INTO employee values(203,'HJK', 'Bangalore',2500);
INSERT INTO employee values(203,'HJK', 'Bangalore',2500)
            *
ERROR at line 1:
ORA-20001: Cannot insert value where id is greater than 100
ORA-06512: at "SYSTEM.INSERT2", line 2
ORA-04088: error during execution of trigger 'SYSTEM.INSERT2'
```

3. Create a trigger that will not allow the user to drop any table

Table :

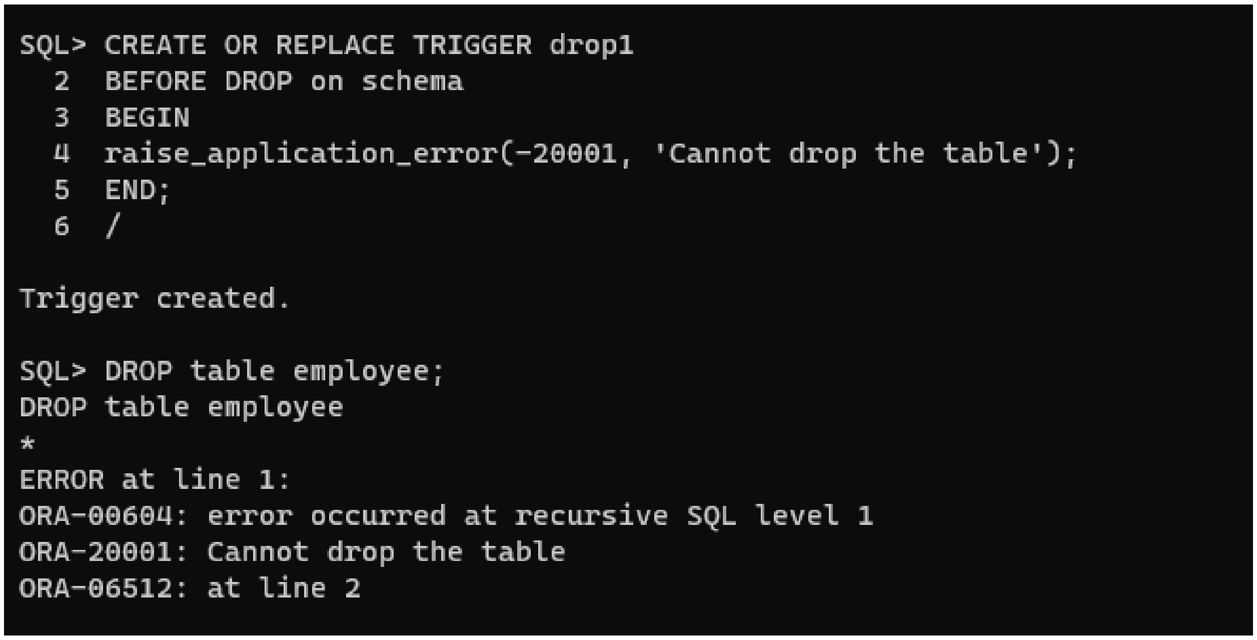
```
 CREATE TABLE employee (
  id NUMBER NOT NULL,
  name VARCHAR2(50) NOT NULL,
  location VARCHAR2(50) NOT NULL,
  salary NUMBER NOT NULL
  );

INSERT INTO employee values(101,'ABC', 'Mumbai',1000);
INSERT INTO employee values(102,'PQR', 'Pune',78000);
```



```
SQL> CREATE TABLE employee (
  2  id NUMBER NOT NULL,
  3  name VARCHAR2(50) NOT NULL,
  4  location VARCHAR2(50) NOT NULL,
  5  salary NUMBER NOT NULL
  6  );

Table created.

SQL> INSERT INTO employee values(101,'ABC', 'Mumbai',1000);

1 row created.

SQL> INSERT INTO employee values(102,'PQR', 'Pune',78000);

1 row created.
```

Code :

```
CREATE OR REPLACE TRIGGER drop1
   BEFORE DROP on schema
   BEGIN
   raise_application_error(-20001, 'Cannot drop the table');
   END;
   /
```



```
SQL> CREATE OR REPLACE TRIGGER drop1
  2  BEFORE DROP on schema
  3  BEGIN
  4  raise_application_error(-20001, 'Cannot drop the table');
  5  END;
  6  /

Trigger created.

SQL> DROP table employee;
DROP table employee
*
ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-20001: Cannot drop the table
ORA-06512: at line 2
```

4. Write a program in PL/SQL to drop a trigger

Table :

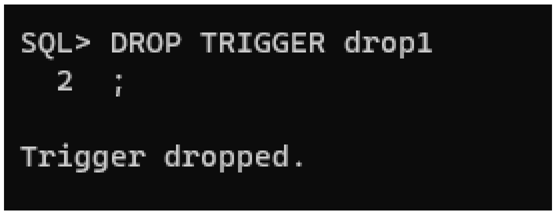
```
 CREATE TABLE employee (
  id NUMBER NOT NULL,
  name VARCHAR2(50) NOT NULL,
  location VARCHAR2(50) NOT NULL,
  salary NUMBER NOT NULL
  );

>INSERT INTO employee values(101,'ABC', 'Mumbai',1000);
>INSERT INTO employee values(102,'PQR', 'Pune',78000);
```

Trigger :

```
CREATE OR REPLACE TRIGGER drop1
    BEFORE DROP on schema
    BEGIN
    raise_application_error(-20001, 'Cannot drop the table');
    END;
    /
```

Code :

```
DROP TRIGGER drop1 ;
```



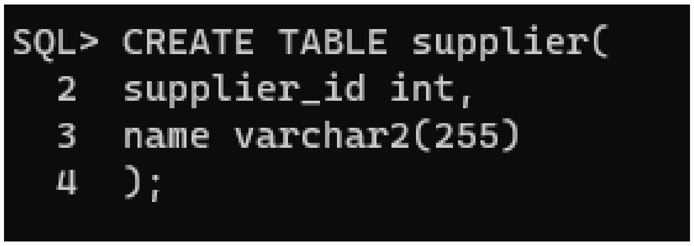
```
SQL> DROP TRIGGER drop1
  2  ;

Trigger dropped.
```

Practical 10

1. Create a supplier table with attribute suplier_id and name. Create a sequence to increment the id by 2, minvalue 1, maxvalue 99

Creating the table :

```
CREATE TABLE supplier(
   supplier_id int,
   name varchar2(255)
   );
```



```
SQL> CREATE TABLE supplier(
  2  supplier_id int,
  3  name varchar2(255)
  4  );
```

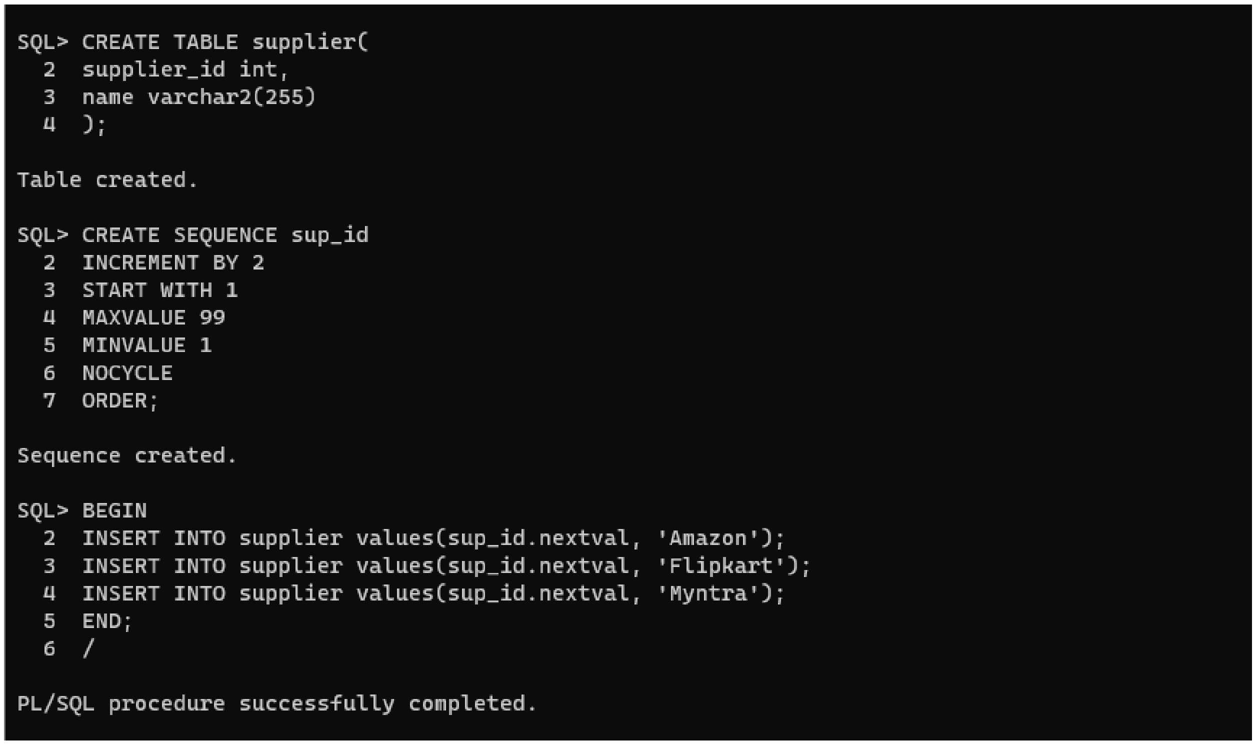
Code :

Sequence :

```
CREATE SEQUENCE sup_id
   INCREMENT BY 2
   START WITH 1
   MAXVALUE 99
   MINVALUE 1
   NOCYCLE
   ORDER;
```

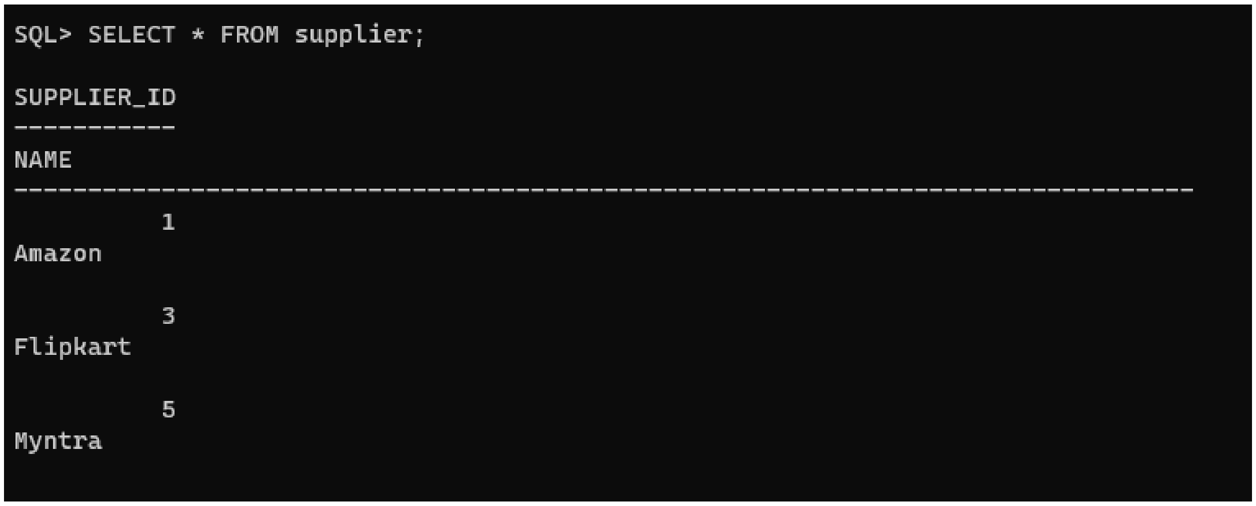
Inserting values :

```
BEGIN
   INSERT INTO supplier values(sup_id.nextval, 'Amazon');
   INSERT INTO supplier values(sup_id.nextval, 'Flipkart');
   INSERT INTO supplier values(sup_id.nextval, 'Myntra');
   END;
   /
```



```
SQL> CREATE TABLE supplier(
supplier_id int,
name varchar2(255)
);

Table created.

SQL> CREATE SEQUENCE sup_id
INCREMENT BY 2
START WITH 1
MAXVALUE 99
MINVALUE 1
NOCYCLE
ORDER;

Sequence created.

SQL> BEGIN
INSERT INTO supplier values(sup_id.nextval, 'Amazon');
INSERT INTO supplier values(sup_id.nextval, 'Flipkart');
INSERT INTO supplier values(sup_id.nextval, 'Myntra');
END;
/

PL/SQL procedure successfully completed.
```

OUTPUT :

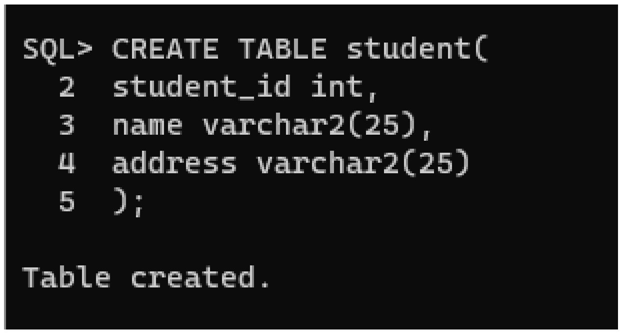


```
SQL> SELECT * FROM supplier;

SUPPLIER_ID
-----------
NAME
--------------------------------------------------------------------------------
          1
Amazon

          3
Flipkart

          5
Myntra
```

2. Create a student table with attribute student roll number and name. Create a sequence to increment the id by 1, minvalue 1, maxvalue 99

Creating the table :

```
CREATE TABLE student(
   student_id int,
   name varchar2(25),
   address varchar2(25)
   );
```



```
SQL> CREATE TABLE student(
  2  student_id int,
  3  name varchar2(25),
  4  address varchar2(25)
  5  );

Table created.
```

Code :

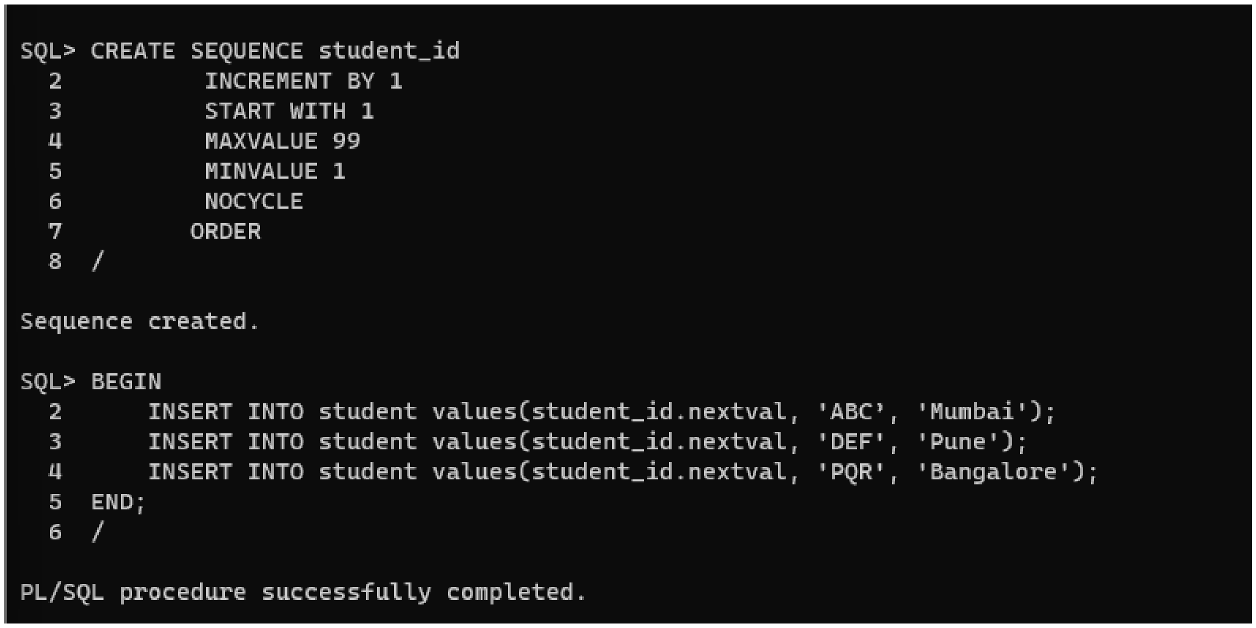
Sequence :

```
 CREATE SEQUENCE student_id
       INCREMENT BY 1
       START WITH 1
       MAXVALUE 99
       MINVALUE 1
       NOCYCLE
      ORDER
/
```

Inserting values :

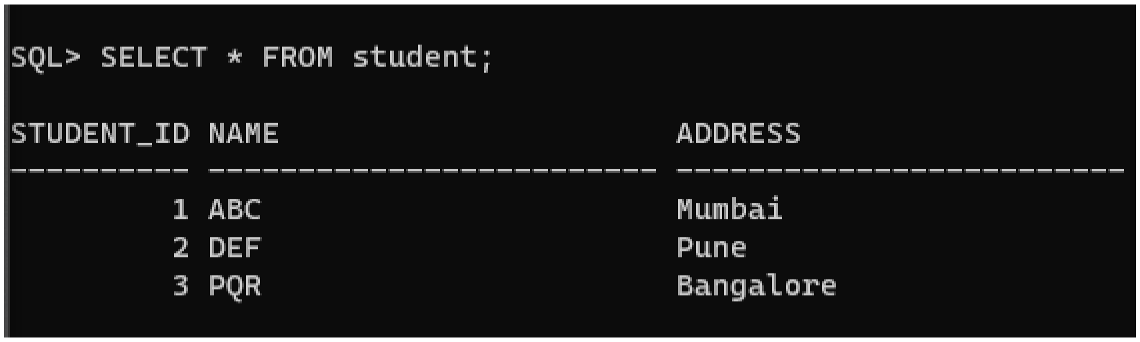
```
BEGIN
   INSERT INTO student values(student_id.nextval, 'ABC', 'Mumbai');
   INSERT INTO student values(student_id.nextval, 'DEF', 'Pune');
   INSERT INTO student values(student_id.nextval, 'PQR', 'Bangalore');
```

END;
/



```
SQL> CREATE SEQUENCE student_id
             INCREMENT BY 1
             START WITH 1
             MAXVALUE 99
             MINVALUE 1
             NOCYCLE
           ORDER
     /

Sequence created.

SQL> BEGIN
         INSERT INTO student values(student_id.nextval, 'ABC', 'Mumbai');
         INSERT INTO student values(student_id.nextval, 'DEF', 'Pune');
         INSERT INTO student values(student_id.nextval, 'PQR', 'Bangalore');
     END;
     /

PL/SQL procedure successfully completed.
```

OUTPUT :



```
SQL> SELECT * FROM student;

STUDENT_ID NAME                      ADDRESS
---------- ------------------------- -------------------------
         1 ABC                       Mumbai
         2 DEF                       Pune
         3 PQR                       Bangalore
```

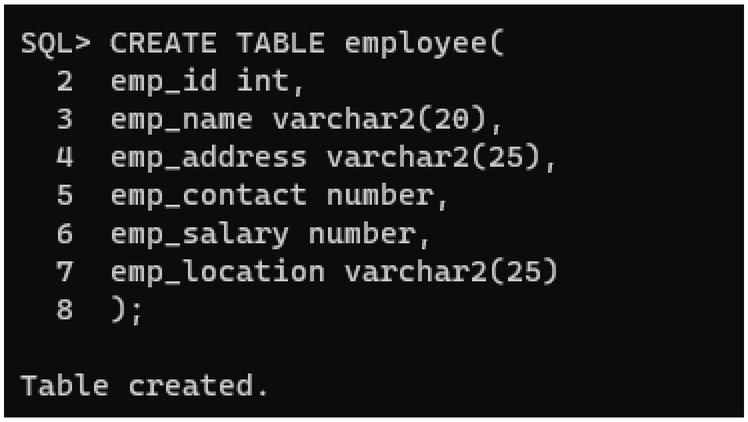
Practical 11

1. Create a table employee with attributes employee id, name, address contact number, salary and location.

- **Insert 5 records in the employee table.**
- **Update the salary as 50000 of an employee whose id is 103**
- **Delete the employee who stays in Ghatkopar**
- **Add column department in the employee table**
- **Add values to the department column in the employee table**
- **Drop table employee**

Creating the table employee :

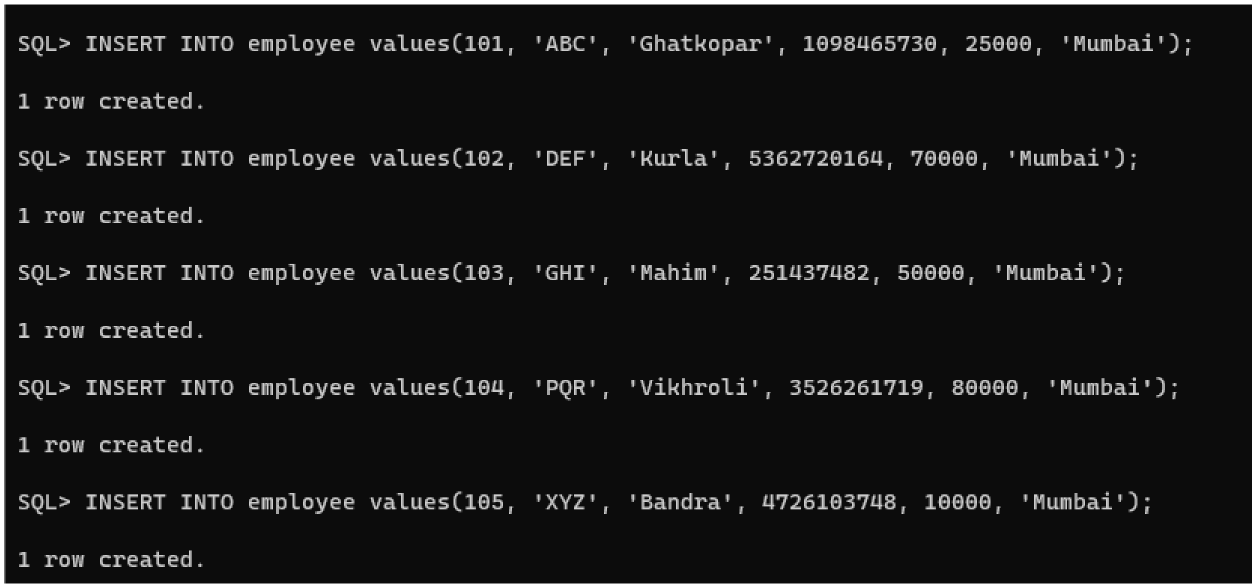
```
CREATE TABLE employee(
  emp_id int,
  emp_name varchar2(20),
  emp_address varchar2(25),
  emp_contact number,
  emp_salary number,
  emp_location varchar2(25)
  );
```



```
SQL> CREATE TABLE employee(
  2  emp_id int,
  3  emp_name varchar2(20),
  4  emp_address varchar2(25),
  5  emp_contact number,
  6  emp_salary number,
  7  emp_location varchar2(25)
  8  );

Table created.
```

a. Inserting 5 records into employee table :

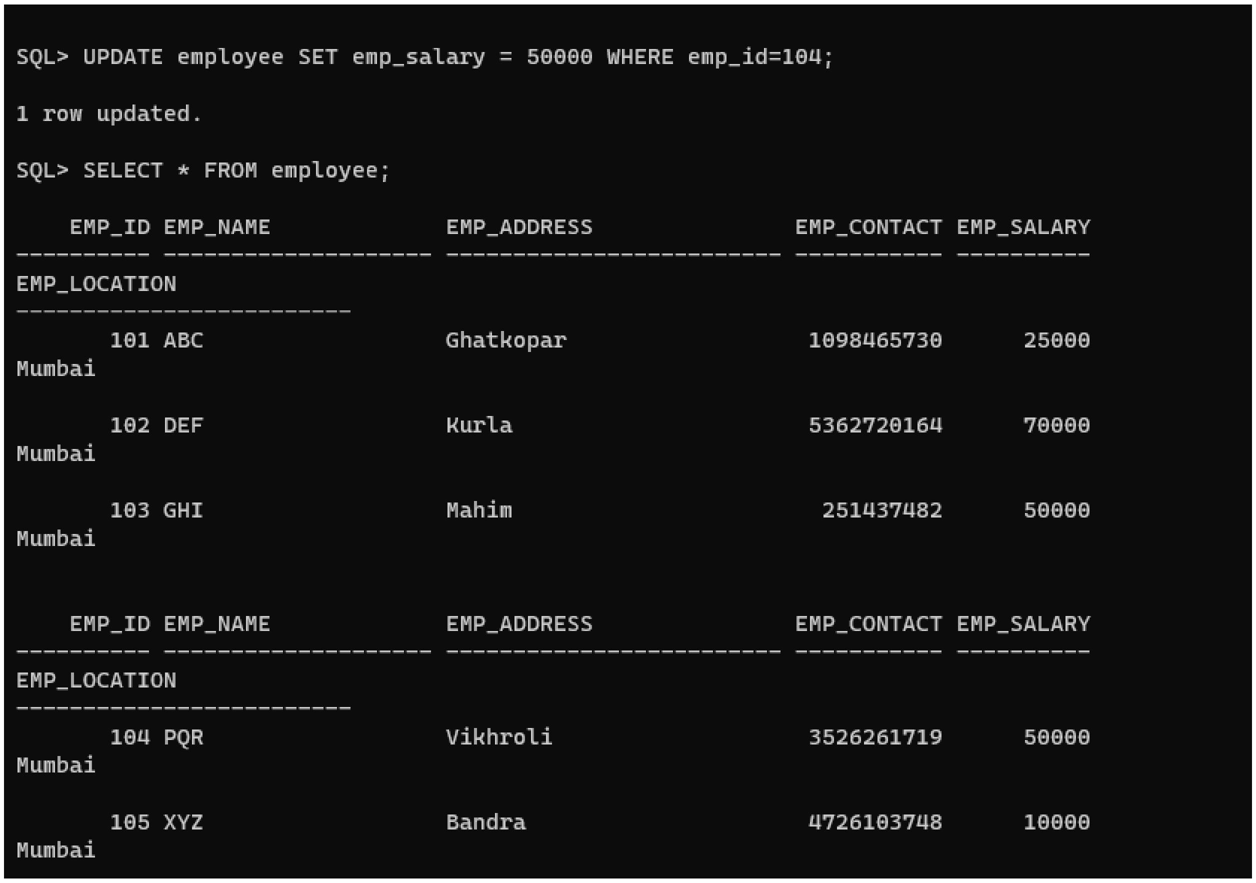
INSERT INTO employee values(101, 'ABC', 'Ghatkopar', 1098465730, 25000, 'Mumbai');
INSERT INTO employee values(102, 'DEF', 'Kurla', 5362720164, 70000, 'Mumbai');
INSERT INTO employee values(103, 'GHI', 'Mahim', 251437482, 50000, 'Mumbai');
INSERT INTO employee values(104, 'PQR', 'Vikhroli', 3526261719, 80000, 'Mumbai');
INSERT INTO employee values(105, 'XYZ', 'Bandra', 4726103748, 10000, 'Mumbai');



```
SQL> INSERT INTO employee values(101, 'ABC', 'Ghatkopar', 1098465730, 25000, 'Mumbai');

1 row created.

SQL> INSERT INTO employee values(102, 'DEF', 'Kurla', 5362720164, 70000, 'Mumbai');

1 row created.

SQL> INSERT INTO employee values(103, 'GHI', 'Mahim', 251437482, 50000, 'Mumbai');

1 row created.

SQL> INSERT INTO employee values(104, 'PQR', 'Vikhroli', 3526261719, 80000, 'Mumbai');

1 row created.

SQL> INSERT INTO employee values(105, 'XYZ', 'Bandra', 4726103748, 10000, 'Mumbai');

1 row created.
```

b. Update the salary as 50000 of an employee whose id is 104

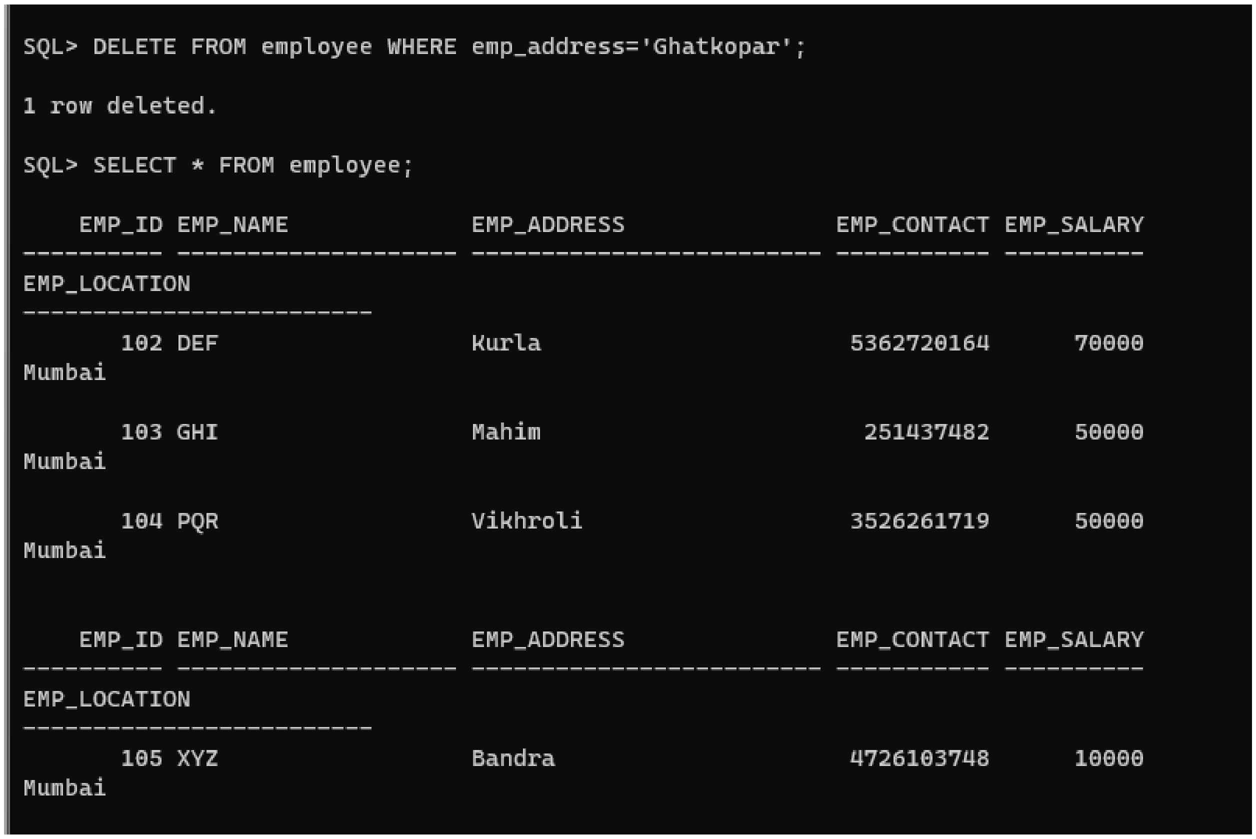
UPDATE employee SET emp_salary = 50000 WHERE emp_id=104;



```
SQL> UPDATE employee SET emp_salary = 50000 WHERE emp_id=104;

1 row updated.

SQL> SELECT * FROM employee;

    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION
-------------------------
       101 ABC                  Ghatkopar                  1098465730      25000
Mumbai

       102 DEF                  Kurla                      5362720164      70000
Mumbai

       103 GHI                  Mahim                       251437482      50000
Mumbai


    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION
-------------------------
       104 PQR                  Vikhroli                   3526261719      50000
Mumbai

       105 XYZ                  Bandra                     4726103748      10000
Mumbai
```

c. Delete the employee who stays in Ghatkopar

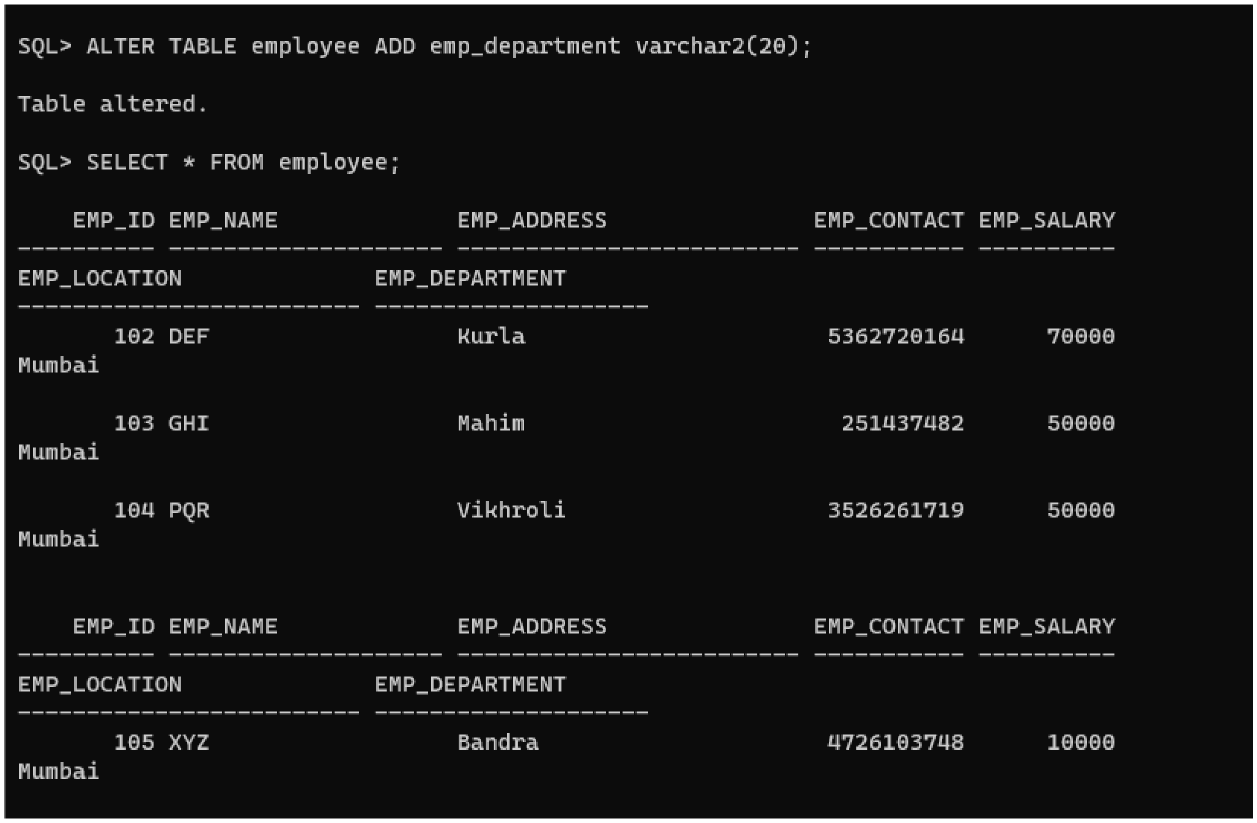
DELETE FROM employee WHERE emp_address='Ghatkopar';



```
SQL> DELETE FROM employee WHERE emp_address='Ghatkopar';

1 row deleted.

SQL> SELECT * FROM employee;

    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION
-------------------------
       102 DEF                  Kurla                      5362720164      70000
Mumbai

       103 GHI                  Mahim                       251437482      50000
Mumbai

       104 PQR                  Vikhroli                   3526261719      50000
Mumbai


    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION
-------------------------
       105 XYZ                  Bandra                     4726103748      10000
Mumbai
```

d. Add column department in the employee table

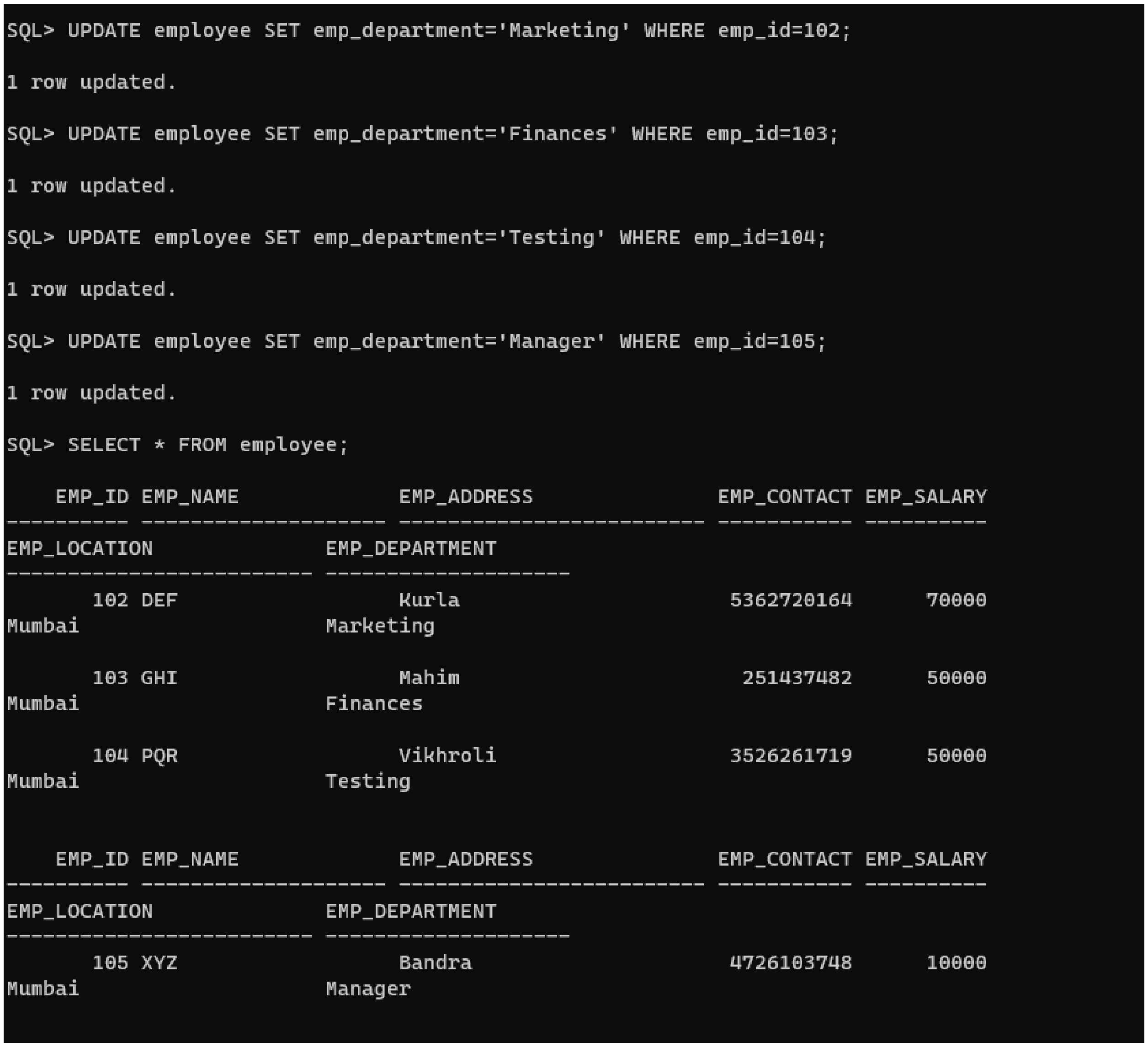
ALTER TABLE employee ADD emp_department varchar2(20);



```
SQL> ALTER TABLE employee ADD emp_department varchar2(20);

Table altered.

SQL> SELECT * FROM employee;

    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION              EMP_DEPARTMENT
------------------------- --------------------
       102 DEF                  Kurla                      5362720164      70000
Mumbai

       103 GHI                  Mahim                       251437482      50000
Mumbai

       104 PQR                  Vikhroli                   3526261719      50000
Mumbai


    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION              EMP_DEPARTMENT
------------------------- --------------------
       105 XYZ                  Bandra                     4726103748      10000
Mumbai
```

e. Update the department column

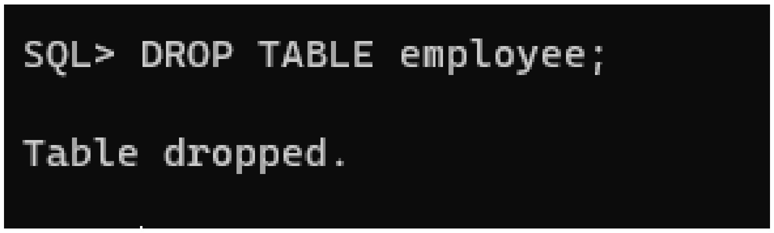
UPDATE employee SET emp_department='Marketing' WHERE emp_id=102;
UPDATE employee SET emp_department='Finances' WHERE emp_id=103;
UPDATE employee SET emp_department='Testing' WHERE emp_id=104;
UPDATE employee SET emp_department='Manager' WHERE emp_id=105;



```
SQL> UPDATE employee SET emp_department='Marketing' WHERE emp_id=102;

1 row updated.

SQL> UPDATE employee SET emp_department='Finances' WHERE emp_id=103;

1 row updated.

SQL> UPDATE employee SET emp_department='Testing' WHERE emp_id=104;

1 row updated.

SQL> UPDATE employee SET emp_department='Manager' WHERE emp_id=105;

1 row updated.

SQL> SELECT * FROM employee;

    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION              EMP_DEPARTMENT
------------------------- --------------------
       102 DEF                  Kurla                      5362720164      70000
Mumbai                    Marketing

       103 GHI                  Mahim                       251437482      50000
Mumbai                    Finances

       104 PQR                  Vikhroli                   3526261719      50000
Mumbai                    Testing


    EMP_ID EMP_NAME             EMP_ADDRESS               EMP_CONTACT EMP_SALARY
---------- -------------------- ------------------------- ----------- ----------
EMP_LOCATION              EMP_DEPARTMENT
------------------------- --------------------
       105 XYZ                  Bandra                     4726103748      10000
Mumbai                    Manager
```

f. Drop table employee

DROP TABLE employee;



```
SQL> DROP TABLE employee;

Table dropped.
```