PL/SQL Programs Record

Submitted by: Alphons baby

Roll no : 10

Subject :ADBMS

The 'Hello World' Example

Program

```
DECLARE
   message  varchar2(20):= 'Hello, World!';
BEGIN
   dbms_output.put_line(message);
END;
```

Output

```
DECLARE
   message  varchar2(20):= 'Hello, World!';
BEGIN
   dbms_output.put_line(message);
END;
```

```
Statement processed.
Hello, World!
```

The PL/SQL Comments

Program

```
DECLARE
   -- variable declaration
   message  varchar2(20):= 'Hello, World!';
BEGIN
   /*
   *  PL/SQL executable statement(s)
```

```
  */
  dbms_output.put_line(message);
END;
```

Output

```
DECLARE
   -- variable declaration
   message  varchar2(20):= 'Hello, World!';
BEGIN
   /*
   *  PL/SQL executable statement(s)
   */
   dbms_output.put_line(message);
END;
```

```
Statement processed.
Hello, World!
```

PL/SQL Numeric Data Types

Program

```
DECLARE
  num1 INTEGER;
  num2 REAL;
  num3 DOUBLE PRECISION;
BEGIN
  null;
END;
```

Output

```
DECLARE
   num1 INTEGER;
   num2 REAL;
   num3 DOUBLE PRECISION;
BEGIN
   null;
END;
```

```
Statement processed.
```

PL/SQL User-Defined Subtypes

Program

```
DECLARE
  SUBTYPE name IS char(20);
  SUBTYPE message IS varchar2(100);
  salutation name;
  greetings message;
BEGIN
  salutation := 'Reader ';
  greetings := 'Welcome to the World of PL/SQL';
  dbms_output.put_line('Hello ' || salutation || greetings);
END;
```

Output

```
DECLARE
   SUBTYPE name IS char(20);
   SUBTYPE message IS varchar2(100);
   salutation name;
   greetings message;
BEGIN
   salutation := 'Reader ';
   greetings := 'Welcome to the World of PL/SQL';
   dbms_output.put_line('Hello ' || salutation || greetings);
END;
```

```
Statement processed.
Hello Reader              Welcome to the World of PL/SQL
```

Variable Declaration in PL/SQL

<u>Program</u>

```
DECLARE
  a integer := 10;
  b integer := 20;
  c integer;
  f real;
BEGIN
  c := a + b;
  dbms_output.put_line('Value of c: ' || c);
  f := 70.0/3.0;
  dbms_output.put_line('Value of f: ' || f);
END;
```

<u>Output</u>

```
DECLARE
   a integer := 10;
   b integer := 20;
   c integer;
   f real;
BEGIN
   c := a + b;
   dbms_output.put_line('Value of c: ' || c);
   f := 70.0/3.0;
   dbms_output.put_line('Value of f: ' || f);
END;
```

```
Statement processed.
Value of c: 30
Value of f: 23.3333333333333333333333333333333333333
```

Variable Scope in PL/SQL

<u>Program</u>

```
DECLARE
  -- Global variables
  num1 number := 95;
  num2 number := 85;
BEGIN
  dbms_output.put_line('Outer Variable num1: ' || num1);
  dbms_output.put_line('Outer Variable num2: ' || num2);
  DECLARE
    -- Local variables
    num1 number := 195;
    num2 number := 185;
  BEGIN
```

```
      dbms_output.put_line('Inner Variable num1: ' || num1);
      dbms_output.put_line('Inner Variable num2: ' || num2);
   END;
END;
```

Output

```
DECLARE
   -- Global variables
   num1 number := 95;
   num2 number := 85;
BEGIN
   dbms_output.put_line('Outer Variable num1: ' || num1);
   dbms_output.put_line('Outer Variable num2: ' || num2);
   DECLARE
      -- Local variables
      num1 number := 195;
      num2 number := 185;
   BEGIN
      dbms_output.put_line('Inner Variable num1: ' || num1);
      dbms_output.put_line('Inner Variable num2: ' || num2);
   END;
END;
```

```
Statement processed.
Outer Variable num1: 95
Outer Variable num2: 85
Inner Variable num1: 195
Inner Variable num2: 185
```

Assigning SQL Query Results to PL/SQL Variables

Program

```
CREATE TABLE CUSTOMERS(
   ID   INT NOT NULL,
   NAME VARCHAR (20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
```

);

```
CREATE TABLE CUSTOMERS(
   ID   INT NOT NULL,
   NAME VARCHAR (20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY   DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

```
Table created.
```

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00);

```sql
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

<u>Program</u>

```
DECLARE
   c_id customers.id%type := 1;
   c_name  customers.name%type;
   c_addr customers.address%type;
   c_sal  customers.salary%type;
BEGIN
   SELECT name, address, salary INTO c_name, c_addr, c_sal
   FROM customers
   WHERE id = c_id;
   dbms_output.put_line
   ('Customer ' ||c_name || ' from ' || c_addr || ' earns ' || c_sal);
END;
```

Output

```
DECLARE
   c_id customers.id%type := 1;
   c_name  customers.name%type;
   c_addr customers.address%type;
   c_sal  customers.salary%type;
BEGIN
   SELECT name, address, salary INTO c_name, c_addr, c_sal
   FROM customers
   WHERE id = c_id;
   dbms_output.put_line
   ('Customer ' ||c_name || ' from ' || c_addr || ' earns ' || c_sal);
END;
```

```
Statement processed.
Customer Ramesh from Ahmedabad                earns 2000
```

Declaring a Constant

Program

PI CONSTANT NUMBER := 3.141592654;

DECLARE

-- constant declaration

pi constant number := 3.141592654;

-- other declarations

radius number(5,2);

dia number(5,2);

circumference number(7, 2);

area number (10, 2);

BEGIN

-- processing

radius := 9.5;

dia := radius * 2;

circumference := 2.0 * pi * radius;

area := pi * radius * radius;

-- output

```
   dbms_output.put_line('Radius: ' || radius);
   dbms_output.put_line('Diameter: ' || dia);
   dbms_output.put_line('Circumference: ' || circumference);
   dbms_output.put_line('Area: ' || area);
END;
```

Output

```
PI CONSTANT NUMBER := 3.141592654;
DECLARE
   -- constant declaration
   pi constant number := 3.141592654;
   -- other declarations
   radius number(5,2);
   dia number(5,2);
   circumference number(7, 2);
   area number (10, 2);
BEGIN
   -- processing
   radius := 9.5;
   dia := radius * 2;
   circumference := 2.0 * pi * radius;
   area := pi * radius * radius;
   -- output
   dbms_output.put_line('Radius: ' || radius);
   dbms_output.put_line('Diameter: ' || dia);
   dbms_output.put_line('Circumference: ' || circumference);
   dbms_output.put_line('Area: ' || area);
END;
```

```
Invalid statement


Statement processed.
Radius: 9.5
Diameter: 19
Circumference: 59.69
Area: 283.53
```

The PL/SQL Literals

Program

```
DECLARE
   message  varchar2(30):= 'That"s tutorialspoint.com!';
BEGIN
   dbms_output.put_line(message);
```

END; ;

Output

```
DECLARE
   message  varchar2(30):= 'That''s tutorialspoint.com!';
BEGIN
   dbms_output.put_line(message);
END;
```

```
Statement processed.
That's tutorialspoint.com!
```

PL/SQL - Arithmetic Operators

Program

```
BEGIN
   dbms_output.put_line( 10 + 5);
   dbms_output.put_line( 10 - 5);
   dbms_output.put_line( 10 * 5);
   dbms_output.put_line( 10 / 5);
   dbms_output.put_line( 10 ** 5);
END; ;
```

Output

```
BEGIN
   dbms_output.put_line( 10 + 5);
   dbms_output.put_line( 10 - 5);
   dbms_output.put_line( 10 * 5);
   dbms_output.put_line( 10 / 5);
   dbms_output.put_line( 10 ** 5);
END;
```

```
Statement processed.
15
5
50
2
100000
```

PL/SQL - Relational Operators

<u>Program</u>

```
DECLARE
  a number (2) := 21;
  b number (2) := 10;
BEGIN
  IF (a = b) then
    dbms_output.put_line('Line 1 - a is equal to b');
  ELSE
    dbms_output.put_line('Line 1 - a is not equal to b');
  END IF;
  IF (a < b) then
    dbms_output.put_line('Line 2 - a is less than b');
  ELSE
```

```
    dbms_output.put_line('Line 2 - a is not less than b');
  END IF;

  IF ( a > b ) THEN
    dbms_output.put_line('Line 3 - a is greater than b');
  ELSE
    dbms_output.put_line('Line 3 - a is not greater than b');
  END IF;
  -- Lets change value of a and b
  a := 5;
  b := 20;
  IF ( a <= b ) THEN
    dbms_output.put_line('Line 4 - a is either equal or less than b');
  END IF;
  IF ( b >= a ) THEN
    dbms_output.put_line('Line 5 - b is either equal or greater than a');
  END IF;
  IF ( a <> b ) THEN
    dbms_output.put_line('Line 6 - a is not equal to b');
  ELSE
    dbms_output.put_line('Line 6 - a is equal to b');
  END IF;
END; ;
```

Output

```
DECLARE
   a number (2) := 21;
   b number (2) := 10;
BEGIN
   IF (a = b) then
      dbms_output.put_line('Line 1 - a is equal to b');
   ELSE
      dbms_output.put_line('Line 1 - a is not equal to b');
   END IF;
   IF (a < b) then
      dbms_output.put_line('Line 2 - a is less than b');
   ELSE
      dbms_output.put_line('Line 2 - a is not less than b');
   END IF;

   IF ( a > b ) THEN
      dbms_output.put_line('Line 3 - a is greater than b');
   ELSE
      dbms_output.put_line('Line 3 - a is not greater than b');
   END IF;
   -- Lets change value of a and b
   a := 5;
   b := 20;
   IF ( a <= b ) THEN
      dbms_output.put_line('Line 4 - a is either equal or less than b');
   END IF;
   IF ( b >= a ) THEN
      dbms_output.put_line('Line 5 - b is either equal or greater than a');
   END IF;
   IF ( a <> b ) THEN
```

```
Statement processed.
15
5
50
2
100000
```

PL/SQL - Comparison Operators

<u>Program</u>

DECLARE

PROCEDURE compare (value varchar2, pattern varchar2) is

BEGIN

  IF value LIKE pattern THEN

    dbms_output.put_line ('True');

```
   ELSE
      dbms_output.put_line ('False');
   END IF;
END;
BEGIN
   compare('Zara Ali', 'Z%A_i');
   compare('Nuha Ali', 'Z%A_i');
END; ;
```

Output

```
DECLARE
PROCEDURE compare (value  varchar2,  pattern varchar2 ) is
BEGIN
   IF value LIKE pattern THEN
      dbms_output.put_line ('True');
   ELSE
      dbms_output.put_line ('False');
   END IF;
END;
BEGIN
   compare('Zara Ali', 'Z%A_i');
   compare('Nuha Ali', 'Z%A_i');
END;
```

```
Statement processed.
True
False
```

Logical Operators in PL/SQL

Program

```
DECLARE
   a boolean := true;
   b boolean := false;
```

```
BEGIN
  IF (a AND b) THEN
    dbms_output.put_line('Line 1 - Condition is true');
  END IF;
  IF (a OR b) THEN
    dbms_output.put_line('Line 2 - Condition is true');
  END IF;
  IF (NOT a) THEN
    dbms_output.put_line('Line 3 - a is not true');
  ELSE
    dbms_output.put_line('Line 3 - a is true');
  END IF;
  IF (NOT b) THEN
    dbms_output.put_line('Line 4 - b is not true');
  ELSE
    dbms_output.put_line('Line 4 - b is true');
  END IF;
END;
```

Output

```
DECLARE
   a boolean := true;
   b boolean := false;
BEGIN
   IF (a AND b) THEN
      dbms_output.put_line('Line 1 - Condition is true');
   END IF;
   IF (a OR b) THEN
      dbms_output.put_line('Line 2 - Condition is true');
   END IF;
   IF (NOT a) THEN
      dbms_output.put_line('Line 3 - a is not true');
   ELSE
      dbms_output.put_line('Line 3 - a is true');
   END IF;
   IF (NOT b) THEN
      dbms_output.put_line('Line 4 - b is not true');
   ELSE
      dbms_output.put_line('Line 4 - b is true');
   END IF;
END;
```

```
Statement processed.
Line 2 - Condition is true
Line 3 - a is true
Line 4 - b is not true
```

PL/SQL Operator Precedence

<u>Program</u>

```
DECLARE

   a number(2) := 20;

   b number(2) := 10;

   c number(2) := 15;

   d number(2) := 5;

   e number(2) ;

BEGIN

   e := (a + b) * c / d;      -- ( 30 * 15 ) / 5

   dbms_output.put_line('Value of (a + b) * c / d is : '|| e );

   e := ((a + b) * c) / d;   -- (30 * 15 ) / 5
```

```
   dbms_output.put_line('Value of ((a + b) * c) / d is  : ' ||  e );

   e := (a + b) * (c / d);   -- (30) * (15/5)

   dbms_output.put_line('Value of (a + b) * (c / d) is  : '||  e );

   e := a + (b * c) / d;     --  20 + (150/5)

   dbms_output.put_line('Value of a + (b * c) / d is  : ' ||  e );

END;
```

Output

```
DECLARE
   a number(2) := 20;
   b number(2) := 10;
   c number(2) := 15;
   d number(2) := 5;
   e number(2) ;
BEGIN
   e := (a + b) * c / d;       -- ( 30 * 15 ) / 5
   dbms_output.put_line('Value of (a + b) * c / d is : '|| e );
   e := ((a + b) * c) / d;    -- (30 * 15 ) / 5
   dbms_output.put_line('Value of ((a + b) * c) / d is  : ' ||  e );
   e := (a + b) * (c / d);    -- (30) * (15/5)
   dbms_output.put_line('Value of (a + b) * (c / d) is  : '||  e );
   e := a + (b * c) / d;      --  20 + (150/5)
   dbms_output.put_line('Value of a + (b * c) / d is  : ' ||  e );
END;
```

```
Statement processed.
Value of (a + b) * c / d is : 90
Value of ((a + b) * c) / d is  : 90
Value of (a + b) * (c / d) is  : 90
Value of a + (b * c) / d is  : 50
```

PL/SQL - Conditions IF-THEN Statement

Program

```
DECLARE
   a number(2) := 10;
BEGIN
   a:= 10;
  -- check the boolean condition using if statement
```

```
  IF( a < 20 ) THEN
    -- if condition is true then print the following
    dbms_output.put_line('a is less than 20 ' );
  END IF;
  dbms_output.put_line('value of a is : ' || a);
END;
```

Output

```
DECLARE
   a number(2) := 10;
BEGIN
   a:= 10;
  -- check the boolean condition using if statement
   IF( a < 20 ) THEN
      -- if condition is true then print the following
      dbms_output.put_line('a is less than 20 ' );
   END IF;
   dbms_output.put_line('value of a is : ' || a);
END;
```

```
Statement processed.
a is less than 20
value of a is : 10
```

PL/SQL - IF-THEN-ELSE Statement

Program

```
DECLARE
  a number(3) := 100;
BEGIN
  -- check the boolean condition using if statement
  IF( a < 20 ) THEN
    -- if condition is true then print the following
    dbms_output.put_line('a is less than 20 ' );
  ELSE
    dbms_output.put_line('a is not less than 20 ' );
```

```
   END IF;
   dbms_output.put_line('value of a is : ' || a);
END;
```

Output

```
DECLARE
   a number(3) := 100;
BEGIN
   -- check the boolean condition using if statement
   IF( a < 20 ) THEN
      -- if condition is true then print the following
      dbms_output.put_line('a is less than 20 ' );
   ELSE
      dbms_output.put_line('a is not less than 20 ' );
   END IF;
   dbms_output.put_line('value of a is : ' || a);
END;
```

```
Statement processed.
a is not less than 20
value of a is : 100
```

PL/SQL - IF-THEN-ELSIF Statement

Program

```
DECLARE
   a number(3) := 100;
BEGIN
   IF ( a = 10 ) THEN
      dbms_output.put_line('Value of a is 10' );
   ELSIF ( a = 20 ) THEN
      dbms_output.put_line('Value of a is 20' );
   ELSIF ( a = 30 ) THEN
      dbms_output.put_line('Value of a is 30' );
   ELSE
       dbms_output.put_line('None of the values is matching');
   END IF;
```

```
   dbms_output.put_line('Exact value of a is: '|| a );
END;
```

Output

```
DECLARE
   a number(3) := 100;
BEGIN
   IF ( a = 10 ) THEN
      dbms_output.put_line('Value of a is 10' );
   ELSIF ( a = 20 ) THEN
      dbms_output.put_line('Value of a is 20' );
   ELSIF ( a = 30 ) THEN
      dbms_output.put_line('Value of a is 30' );
   ELSE
       dbms_output.put_line('None of the values is matching');
   END IF;
   dbms_output.put_line('Exact value of a is: '|| a );
END;
```

```
Statement processed.
None of the values is matching
Exact value of a is: 100
```

PL/SQL - CASE Statement

Program

```
DECLARE
   grade char(1) := 'A';
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('You passed');
      when 'F' then dbms_output.put_line('Better try again');
      else dbms_output.put_line('No such grade');
   END CASE;
END;
```

Output

```
DECLARE
   grade char(1) := 'A';
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('You passed');
      when 'F' then dbms_output.put_line('Better try again');
      else dbms_output.put_line('No such grade');
   END CASE;
END;
```

```
Statement processed.
Excellent
```

PL/SQL - Searched CASE Statement

Program

```
DECLARE
  grade char(1) := 'B';
BEGIN
  case
    when grade = 'A' then dbms_output.put_line('Excellent');
    when grade = 'B' then dbms_output.put_line('Very good');
    when grade = 'C' then dbms_output.put_line('Well done');
    when grade = 'D' then dbms_output.put_line('You passed');
    when grade = 'F' then dbms_output.put_line('Better try again');
    else dbms_output.put_line('No such grade');
  end case;
END;
```

Output

```
DECLARE
   grade char(1) := 'B';
BEGIN
   case
      when grade = 'A' then dbms_output.put_line('Excellent');
      when grade = 'B' then dbms_output.put_line('Very good');
      when grade = 'C' then dbms_output.put_line('Well done');
      when grade = 'D' then dbms_output.put_line('You passed');
      when grade = 'F' then dbms_output.put_line('Better try again');
      else dbms_output.put_line('No such grade');
   end case;
END;
```

```
Statement processed.
Very good
```

PL/SQL - Nested IF-THEN-ELSE Statements

Program

```
DECLARE
  a number(3) := 100;
  b number(3) := 200;
BEGIN
  -- check the boolean condition
  IF( a = 100 ) THEN
  -- if condition is true then check the following
    IF( b = 200 ) THEN
    -- if condition is true then print the following
    dbms_output.put_line('Value of a is 100 and b is 200' );
    END IF;
  END IF;
  dbms_output.put_line('Exact value of a is : ' || a );
  dbms_output.put_line('Exact value of b is : ' || b );
END;
```

Output

```
DECLARE
   a number(3) := 100;
   b number(3) := 200;
BEGIN
   -- check the boolean condition
   IF( a = 100 ) THEN
   -- if condition is true then check the following
      IF( b = 200 ) THEN
      -- if condition is true then print the following
      dbms_output.put_line('Value of a is 100 and b is 200' );
      END IF;
   END IF;
   dbms_output.put_line('Exact value of a is : ' || a );
   dbms_output.put_line('Exact value of b is : ' || b );
END;
```

```
Statement processed.
Value of a is 100 and b is 200
Exact value of a is : 100
Exact value of b is : 200
```

PL/SQL - Basic Loop Statement

Program

```
DECLARE
  x number := 10;
BEGIN
  LOOP
    dbms_output.put_line(x);
    x := x + 10;
    exit WHEN x > 50;
  END LOOP;
  -- after exit, control resumes here
  dbms_output.put_line('After Exit x is: ' || x);
END;
```

Output

```
DECLARE
   x number := 10;
BEGIN
   LOOP
      dbms_output.put_line(x);
      x := x + 10;
      exit WHEN x > 50;
   END LOOP;
   -- after exit, control resumes here
   dbms_output.put_line('After Exit x is: ' || x);
END;
```

```
Statement processed.
10
20
30
40
50
After Exit x is: 60
```

PL/SQL - WHILE LOOP Statement

Program

```
DECLARE
  a number(2) := 10;
BEGIN
  WHILE a < 20 LOOP
    dbms_output.put_line('value of a: ' || a);
    a := a + 1;
  END LOOP;
END;
```

Output

```
DECLARE
   a number(2) := 10;
BEGIN
   WHILE a < 20 LOOP
      dbms_output.put_line('value of a: ' || a);
      a := a + 1;
   END LOOP;
END;
```

```
Statement processed.
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

PL/SQL - FOR LOOP Statement

Program

```
DECLARE
  a number(2);
BEGIN
  FOR a in 10 .. 20 LOOP
    dbms_output.put_line('value of a: ' || a);
 END LOOP;
END;
```

Output

```
DECLARE
   a number(2);
BEGIN
   FOR a in 10 .. 20 LOOP
      dbms_output.put_line('value of a: ' || a);
  END LOOP;
END;
```

```
Statement processed.
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```

PL/SQL - Nested Loops

<u>Program</u>

```
DECLARE
  i number(3);
  j number(3);
BEGIN
  i := 2;
  LOOP
    j:= 2;
    LOOP
      exit WHEN ((mod(i, j) = 0) or (j = i));
      j := j +1;
    END LOOP;
  IF (j = i ) THEN
    dbms_output.put_line(i || ' is prime');
  END IF;
  i := i + 1;
```

```
   exit WHEN i = 50;
   END LOOP;
END;
/
```

Output

```
DECLARE
   i number(3);
   j number(3);
BEGIN
   i := 2;
   LOOP
      j:= 2;
      LOOP
         exit WHEN ((mod(i, j) = 0) or (j = i));
         j := j +1;
      END LOOP;
   IF (j = i ) THEN
      dbms_output.put_line(i || ' is prime');
   END IF;
   i := i + 1;
   exit WHEN i = 50;
   END LOOP;
END;
/
```

```
Statement processed.
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
```

Labeling a PL/SQL Loop

Program

```
DECLARE
   i number(1);
   j number(1);
BEGIN
```

```
   << outer_loop >>
   FOR i IN 1..3 LOOP
      << inner_loop >>
      FOR j IN 1..3 LOOP
         dbms_output.put_line('i is: '|| i || ' and j is: ' || j);
      END loop inner_loop;
   END loop outer_loop;
END;
/
```

Output

```
DECLARE
   i number(1);
   j number(1);
BEGIN
   << outer_loop >>
   FOR i IN 1..3 LOOP
      << inner_loop >>
      FOR j IN 1..3 LOOP
         dbms_output.put_line('i is: '|| i || ' and j is: ' || j);
      END loop inner_loop;
   END loop outer_loop;
END;
/
```

```
Statement processed.
i is: 1 and j is: 1
i is: 1 and j is: 2
i is: 1 and j is: 3
i is: 2 and j is: 1
i is: 2 and j is: 2
i is: 2 and j is: 3
i is: 3 and j is: 1
i is: 3 and j is: 2
i is: 3 and j is: 3
```

PL/SQL - EXIT Statement

Program

```
DECLARE
   a number(2) := 10;
BEGIN
```

```
  -- while loop execution
  WHILE a < 20 LOOP
    dbms_output.put_line ('value of a: ' || a);
    a := a + 1;
    IF a > 15 THEN
      -- terminate the loop using the exit statement
      EXIT;
    END IF;
  END LOOP;
END;
/
```

<u>Output</u>

```
DECLARE
   a number(2) := 10;
BEGIN
   -- while loop execution
   WHILE a < 20 LOOP
      dbms_output.put_line ('value of a: ' || a);
      a := a + 1;
      IF a > 15 THEN
         -- terminate the loop using the exit statement
         EXIT;
      END IF;
   END LOOP;
END;
/
```

```
Statement processed.
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

PL/SQL - CONTINUE Statement

Program

```
DECLARE
  a number(2) := 10;
BEGIN
  -- while loop execution
  WHILE a < 20 LOOP
    dbms_output.put_line ('value of a: ' || a);
    a := a + 1;
    IF a = 15 THEN
      -- skip the loop using the CONTINUE statement
      a := a + 1;
      CONTINUE;
    END IF;
  END LOOP;
END;
/
```

Output

```
DECLARE
   a number(2) := 10;
BEGIN
   -- while loop execution
   WHILE a < 20 LOOP
      dbms_output.put_line ('value of a: ' || a);
      a := a + 1;
      IF a = 15 THEN
         -- skip the loop using the CONTINUE statement
         a := a + 1;
         CONTINUE;
      END IF;
   END LOOP;
END;
/
```

```
Statement processed.
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

PL/SQL - GOTO Statement

Program

```
DECLARE
  a number(2) := 10;
BEGIN
  <<loopstart>>
  -- while loop execution
  WHILE a < 20 LOOP
  dbms_output.put_line ('value of a: ' || a);
    a := a + 1;
    IF a = 15 THEN
      a := a + 1;
      GOTO loopstart;
    END IF;
```

```
   END LOOP;
END;
/
```

Output

```
DECLARE
   a number(2) := 10;
BEGIN
   <<loopstart>>
   -- while loop execution
   WHILE a < 20 LOOP
   dbms_output.put_line ('value of a: ' || a);
      a := a + 1;
      IF a = 15 THEN
         a := a + 1;
         GOTO loopstart;
      END IF;
   END LOOP;
END;
/
```

```
Statement processed.
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Declaring String Variables

Program

```
DECLARE
   name varchar2(20);
   company varchar2(30);
   introduction clob;
   choice char(1);
BEGIN
   name := 'John Smith';
   company := 'Infotech';
```

```
  introduction := ' Hello! I"m John Smith from Infotech.';
  choice := 'y';
  IF choice = 'y' THEN
    dbms_output.put_line(name);
    dbms_output.put_line(company);
    dbms_output.put_line(introduction);
  END IF;
END;
/
```

Output

```
DECLARE
   name varchar2(20);
   company varchar2(30);
   introduction clob;
   choice char(1);
BEGIN
   name := 'John Smith';
   company := 'Infotech';
   introduction := ' Hello! I''m John Smith from Infotech.';
   choice := 'y';
   IF choice = 'y' THEN
      dbms_output.put_line(name);
      dbms_output.put_line(company);
      dbms_output.put_line(introduction);
   END IF;
END;
/
```

```
Statement processed.
John Smith
Infotech
```

PL/SQL String Functions and Operators

Program

```
DECLARE
  greetings varchar2(11) := 'hello world';
BEGIN
  dbms_output.put_line(UPPER(greetings));
```

```
  dbms_output.put_line(LOWER(greetings));

  dbms_output.put_line(INITCAP(greetings));

  /* retrieve the first character in the string */
  dbms_output.put_line ( SUBSTR (greetings, 1, 1));

  /* retrieve the last character in the string */
  dbms_output.put_line ( SUBSTR (greetings, -1, 1));

  /* retrieve five characters,
     starting from the seventh position. */
  dbms_output.put_line ( SUBSTR (greetings, 7, 5));

  /* retrieve the remainder of the string,
     starting from the second position. */
  dbms_output.put_line ( SUBSTR (greetings, 2));

  /* find the location of the first "e" */
  dbms_output.put_line ( INSTR (greetings, 'e'));
END;
/
```

Output

```
DECLARE
   greetings varchar2(11) := 'hello world';
BEGIN
   dbms_output.put_line(UPPER(greetings));

   dbms_output.put_line(LOWER(greetings));

   dbms_output.put_line(INITCAP(greetings));

   /* retrieve the first character in the string */
   dbms_output.put_line ( SUBSTR (greetings, 1, 1));

   /* retrieve the last character in the string */
   dbms_output.put_line ( SUBSTR (greetings, -1, 1));

   /* retrieve five characters,
      starting from the seventh position. */
   dbms_output.put_line ( SUBSTR (greetings, 7, 5));

   /* retrieve the remainder of the string,
      starting from the second position. */
   dbms_output.put_line ( SUBSTR (greetings, 2));

   /* find the location of the first "e" */
   dbms_output.put_line ( INSTR (greetings, 'e'));
END;
/
```

```
Statement processed.
HELLO WORLD
hello world
Hello World
h
```

PL/SQL – Arrays

<u>Program</u>

```
DECLARE
   type namesarray IS VARRAY(5) OF VARCHAR2(10);
   type grades IS VARRAY(5) OF INTEGER;
   names namesarray;
   marks grades;
   total integer;
BEGIN
   names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
   marks:= grades(98, 97, 78, 87, 92);
   total := names.count;
```

```
   dbms_output.put_line('Total '|| total || ' Students');
   FOR i in 1 .. total LOOP
      dbms_output.put_line('Student: ' || names(i) || '
      Marks: ' || marks(i));
   END LOOP;
END;
/
```

Output

```
DECLARE
   type namesarray IS VARRAY(5) OF VARCHAR2(10);
   type grades IS VARRAY(5) OF INTEGER;
   names namesarray;
   marks grades;
   total integer;
BEGIN
   names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
   marks:= grades(98, 97, 78, 87, 92);
   total := names.count;
   dbms_output.put_line('Total '|| total || ' Students');
   FOR i in 1 .. total LOOP
      dbms_output.put_line('Student: ' || names(i) || '
      Marks: ' || marks(i));
   END LOOP;
END;
/
```

```
Statement processed.
Total 5 Students
Student: Kavita
      Marks: 98
Student: Pritam
      Marks: 97
Student: Ayan
      Marks: 78
Student: Rishav
      Marks: 87
Student: Aziz
      Marks: 92
```

PL/SQL – Procedures

Program

```
DECLARE
   a number;
```

```
   b number;
   c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
   IF x < y THEN
      z:= x;
   ELSE
      z:= y;
   END IF;
END;
BEGIN
   a:= 23;
   b:= 45;
   findMin(a, b, c);
   dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END;
/
```

Output

```
DECLARE
   a number;
   b number;
   c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
   IF x < y THEN
      z:= x;
   ELSE
      z:= y;
   END IF;
END;
BEGIN
   a:= 23;
   b:= 45;
   findMin(a, b, c);
   dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END;
/
```

```
Statement processed.
 Minimum of (23, 45) : 23
```

PL/SQL – Functions

<u>Program</u>

```
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;
BEGIN
  a:= 23;
  b:= 45;
  c := findMax(a, b);
  dbms_output.put_line(' Maximum of (23,45): ' || c);
END;
/
```

<u>Output</u>

```
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;
BEGIN
   a:= 23;
   b:= 45;
   c := findMax(a, b);
   dbms_output.put_line(' Maximum of (23,45): ' || c);
END;
/
```

```
Statement processed.
 Maximum of (23,45): 45
```

PL/SQL Recursive Functions

Program

```
DECLARE
  num number;
  factorial number;

FUNCTION fact(x number)
RETURN number
IS
  f number;
BEGIN
  IF x=0 THEN
    f := 1;
  ELSE
    f := x * fact(x-1);
  END IF;
RETURN f;
END;

BEGIN
  num:= 6;
  factorial := fact(num);
  dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
/
```

Output

```
DECLARE
   num number;
   factorial number;

FUNCTION fact(x number)
RETURN number
IS
   f number;
BEGIN
   IF x=0 THEN
      f := 1;
   ELSE
      f := x * fact(x-1);
   END IF;
RETURN f;
END;

BEGIN
   num:= 6;
   factorial := fact(num);
   dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
/
```

```
Statement processed.
 Factorial 6 is 720
```

PL/SQL – Cursors

<u>Program</u>

```
DECLARE
  c_id customers.id%type;
  c_name customers.name%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  LOOP
  FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
```

```
   CLOSE c_customers;
END;
/
```

Output

```
DECLARE
   c_id customers.id%type;
   c_name customers.name%type;
   c_addr customers.address%type;
   CURSOR c_customers is
      SELECT id, name, address FROM customers;
BEGIN
   OPEN c_customers;
   LOOP
   FETCH c_customers into c_id, c_name, c_addr;
      EXIT WHEN c_customers%notfound;
      dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
   END LOOP;
   CLOSE c_customers;
END;
/
```

```
Statement processed.
1 Ramesh Ahmedabad
2 Khilan Delhi
3 kaushik Kota
4 Chaitali Mumbai
5 Hardik Bhopal
6 Komal MP
```

PL/SQL – Records

Program

```
DECLARE
   customer_rec customers%rowtype;
BEGIN
   SELECT * into customer_rec
   FROM customers
   WHERE id = 5;
   dbms_output.put_line('Customer ID: ' || customer_rec.id);
   dbms_output.put_line('Customer Name: ' || customer_rec.name);
```

```
   dbms_output.put_line('Customer Address: ' || customer_rec.address);
   dbms_output.put_line('Customer Salary: ' || customer_rec.salary);
END;
/
```

Output

```
DECLARE
   customer_rec customers%rowtype;
BEGIN
   SELECT * into customer_rec
   FROM customers
   WHERE id = 5;
   dbms_output.put_line('Customer ID: ' || customer_rec.id);
   dbms_output.put_line('Customer Name: ' || customer_rec.name);
   dbms_output.put_line('Customer Address: ' ||_customer_rec.address);
   dbms_output.put_line('Customer Salary: ' || customer_rec.salary);
END;
/
```

```
Statement processed.
Customer ID: 5
Customer Name: Hardik
Customer Address: Bhopal
Customer Salary: 8500
```

PL/SQL – Exceptions

Program

```
DECLARE
   c_id customers.id%type := 8;
   c_name customerS.Name%type;
   c_addr customers.address%type;
BEGIN
   SELECT  name, address INTO  c_name, c_addr
   FROM customers
   WHERE id = c_id;
   DBMS_OUTPUT.PUT_LINE ('Name: '||  c_name);
```

```
   DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);

EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('No such customer!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
/
```

Output

```
DECLARE
   c_id customers.id%type := 8;
   c_name customerS.Name%type;
   c_addr customers.address%type;
BEGIN
   SELECT  name, address INTO  c_name, c_addr
   FROM customers
   WHERE id = c_id;
   DBMS_OUTPUT.PUT_LINE ('Name: '||  c_name);
   DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);

EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('No such customer!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
/
```

```
Statement processed.
No such customer!
```

PL/SQL – Triggers

Program

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON customers
```

```
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff := :NEW.salary  - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

Output

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

```
Trigger created.
```

PL/SQL - Packages

Program

```
CREATE OR REPLACE PACKAGE c_package AS
   -- Adds a customer
   PROCEDURE addCustomer(c_id   customers.id%type,
   c_name customers.Name%type,
```

```
   c_age  customers.age%type,
   c_addr customers.address%type,
   c_sal  customers.salary%type);

   -- Removes a customer
   PROCEDURE delCustomer(c_id  customers.id%TYPE);
   --Lists all customers
   PROCEDURE listCustomer;

END c_package;
/
```

Output

```
CREATE OR REPLACE PACKAGE c_package AS
   -- Adds a customer
   PROCEDURE addCustomer(c_id   customers.id%type,
   c_name customers.Name%type,
   c_age  customers.age%type,
   c_addr customers.address%type,
   c_sal  customers.salary%type);

   -- Removes a customer
   PROCEDURE delCustomer(c_id  customers.id%TYPE);
   --Lists all customers
   PROCEDURE listCustomer;

END c_package;
/
```

```
Package created.
```

PL/SQL - Collections

Program

```
DECLARE
```

```
  TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
  salary_list salary;
  name   VARCHAR2(20);
BEGIN
  -- adding elements to the table
  salary_list('Rajnish') := 62000;
  salary_list('Minakshi') := 75000;
  salary_list('Martin') := 100000;
  salary_list('James') := 78000;

  -- printing the table
  name := salary_list.FIRST;
  WHILE name IS NOT null LOOP
    dbms_output.put_line
    ('Salary of ' || name || ' is ' || TO_CHAR(salary_list(name)));
    name := salary_list.NEXT(name);
  END LOOP;
END;
/
```

Output

```
DECLARE
   TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
   salary_list salary;
   name   VARCHAR2(20);
BEGIN
   -- adding elements to the table
   salary_list('Rajnish') := 62000;
   salary_list('Minakshi') := 75000;
   salary_list('Martin') := 100000;
   salary_list('James') := 78000;

   -- printing the table
   name := salary_list.FIRST; _
   WHILE name IS NOT null LOOP
      dbms_output.put_line
      ('Salary of ' || name || ' is ' || TO_CHAR(salary_list(name)));
      name := salary_list.NEXT(name);
   END LOOP;
END;
/
```

```
Statement processed.
Salary of James is 78000
Salary of Martin is 100000
Salary of Minakshi is 75000
Salary of Rajnish is 62000
```

PL/SQL - Transactions

Program

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (3, 'kaushik', 23, 'Kota', 2000.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00);

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)

VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00);

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );

COMMIT;
PDATE CUSTOMERS
SET SALARY = SALARY + 1000;
ROLLBACK TO sav1;

UPDATE CUSTOMERS
SET SALARY = SALARY + 1000
WHERE ID = 7;
UPDATE CUSTOMERS
SET SALARY = SALARY + 1000
WHERE ID = 8;
```

Output

```sql

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );

COMMIT;
PDATE CUSTOMERS
SET SALARY = SALARY + 1000;
ROLLBACK TO sav1;

UPDATE CUSTOMERS
SET SALARY = SALARY + 1000
WHERE ID = 7;
UPDATE CUSTOMERS
SET SALARY = SALARY + 1000
WHERE ID = 8;
```

PL/SQL - Date & Time

Program

SELECT SYSDATE FROM DUAL;

SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY HH:MI:SS') FROM DUAL;

SELECT ADD_MONTHS(SYSDATE, 5) FROM DUAL;

SELECT LOCALTIMESTAMP FROM DUAL;

Output

```
SELECT SYSDATE FROM DUAL;
SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY HH:MI:SS') FROM DUAL;
SELECT ADD_MONTHS(SYSDATE, 5) FROM DUAL;
SELECT LOCALTIMESTAMP FROM DUAL; _
```

SYSDATE
05-OCT-21

Download CSV

TO_CHAR(CURRENT_DATE,'DD-MM-YYYYHH:MI:SS')
04-10-2021 11:25:42

Download CSV

ADD_MONTHS(SYSDATE,5)
05-MAR-22

Download CSV

LOCALTIMESTAMP
04-OCT-21 11.25.46.712915 PM

Download CSV

PL/SQL - DBMS Output

Program

```
DECLARE
  lines dbms_output.chararr;
  num_lines number;
BEGIN
  -- enable the buffer with default size 20000
  dbms_output.enable;

  dbms_output.put_line('Hello Reader!');
  dbms_output.put_line('Hope you have enjoyed the tutorials!');
```

```
  dbms_output.put_line('Have a great time exploring pl/sql!');

  num_lines := 3;

  dbms_output.get_lines(lines, num_lines);

  FOR i IN 1..num_lines LOOP
    dbms_output.put_line(lines(i));
  END LOOP;
END;
/
```

Output

```
DECLARE
   lines dbms_output.chararr;
   num_lines number;
BEGIN
   -- enable the buffer with default size 20000
   dbms_output.enable;

   dbms_output.put_line('Hello Reader!');
   dbms_output.put_line('Hope you have enjoyed the tutorials!');
   dbms_output.put_line('Have a great time exploring pl/sql!');

   num_lines := 3;

   dbms_output.get_lines(lines, num_lines);

   FOR i IN 1..num_lines LOOP
      dbms_output.put_line(lines(i));
   END LOOP;
END;
/
```

```
Statement processed.
Hello Reader!
Hope you have enjoyed the tutorials!
Have a great time exploring pl/sql!
```

PL/SQL - Object Oriented

<u>Program</u>

```
CREATE OR REPLACE TYPE rectangle AS OBJECT
(length number,
 width number,
 member function enlarge( inc number) return rectangle,
 member procedure display,
 map member function measure return number
);
/
```

Output

```
CREATE OR REPLACE TYPE rectangle AS OBJECT
(length number,
 width number,
 member function enlarge( inc number) return rectangle,
 member procedure display,
 map member function measure return number
);
/
```

```
Type created.
```

Program

```
CREATE OR REPLACE TYPE BODY rectangle AS
  MEMBER FUNCTION enlarge(inc number) return rectangle IS
  BEGIN
    return rectangle(self.length + inc, self.width + inc);
  END enlarge;
  MEMBER PROCEDURE display IS
  BEGIN
    dbms_output.put_line('Length: '|| length);
    dbms_output.put_line('Width: '|| width);
  END display;
  MAP MEMBER FUNCTION measure return number IS
  BEGIN
    return (sqrt(length*length + width*width));
  END measure;
END;
/
```

Output

```
CREATE OR REPLACE TYPE BODY rectangle AS
   MEMBER FUNCTION enlarge(inc number) return rectangle IS
   BEGIN
      return rectangle(self.length + inc, self.width + inc);
   END enlarge;
   MEMBER PROCEDURE display IS
   BEGIN
      dbms_output.put_line('Length: '|| length);
      dbms_output.put_line('Width: '|| width);
   END display;
   MAP MEMBER FUNCTION measure return number IS
   BEGIN
      return (sqrt(length*length + width*width));
   END measure;
END;
/ _
```

```
Type created.
```

Program

```
DECLARE
  r1 rectangle;
  r2 rectangle;
  r3 rectangle;
  inc_factor number := 5;
BEGIN
  r1 := rectangle(3, 4);
  r2 := rectangle(5, 7);
  r3 := r1.enlarge(inc_factor);
  r3.display;
  IF (r1 > r2) THEN -- calling measure function
    r1.display;
  ELSE
    r2.display;
```

```
   END IF;
END;
/
```

<u>Output</u>

```
DECLARE
   r1 rectangle;
   r2 rectangle;
   r3 rectangle;
   inc_factor number := 5;
BEGIN
   r1 := rectangle(3, 4);
   r2 := rectangle(5, 7);
   r3 := r1.enlarge(inc_factor);
   r3.display;
   IF (r1 > r2) THEN -- calling measure function
      r1.display;
   ELSE
      r2.display;
   END IF;
END;
/
```

```
Statement processed.
Length: 8
Width: 9
Length: 5
Width: 7
```