

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

1	DECLARE	
2	message varchar2(20):= 'Hello, World!';	
3	BEGIN	
4	dbms_output.put_line(message);	
5	END;	
6		

Statement processed. Hello, World!

The PL/SQL Comments

Program

DECLARE

 -- variable declaration

 message varchar2(20):= 'Hello, World!';

BEGIN

 /*

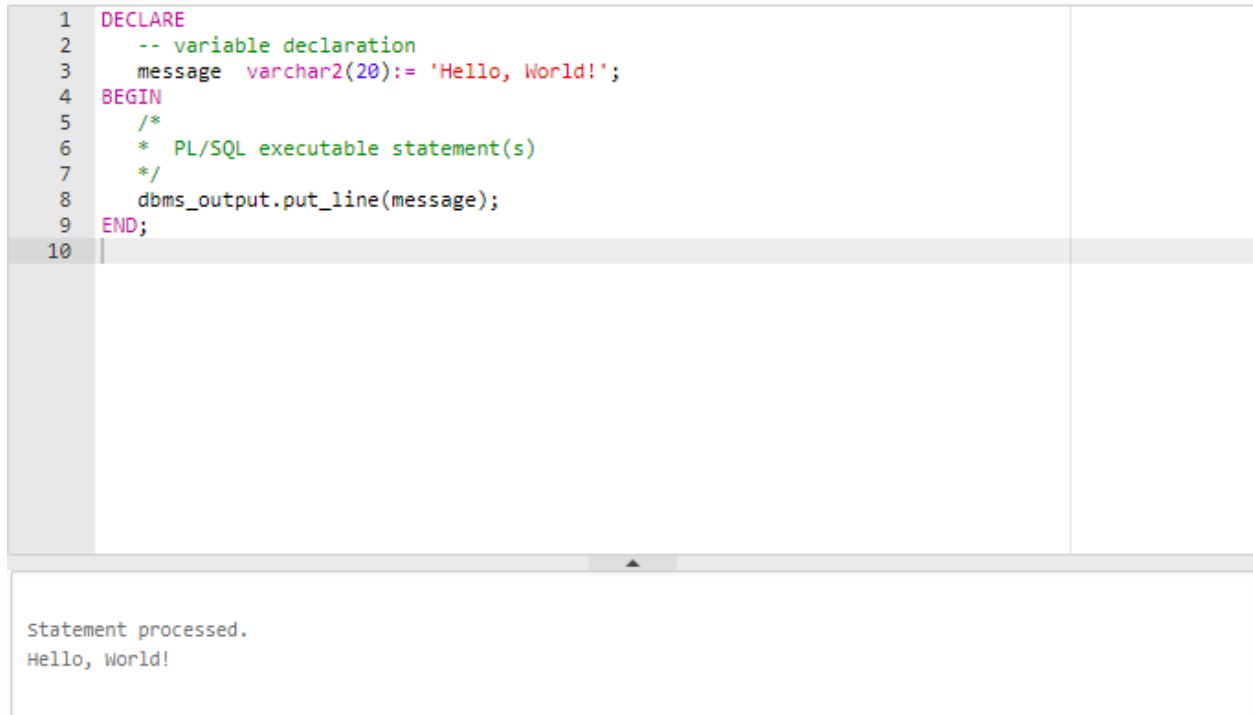
 * PL/SQL executable statement(s)

```
*/
```

```
dbms_output.put_line(message);
```

```
END;
```

Output



The screenshot shows a SQL IDE with a code editor and an output window. The code editor contains a PL/SQL program with line numbers 1 through 10. The output window shows the result of the program execution.

```
1 DECLARE
2   -- variable declaration
3   message varchar2(20) := 'Hello, World!';
4 BEGIN
5   /*
6    * PL/SQL executable statement(s)
7    */
8   dbms_output.put_line(message);
9 END;
10
```

statement processed.
Hello, World!

PL/SQL Numeric Data Types

Program

```
DECLARE
```

```
num1 INTEGER;
```

```
num2 REAL;
```

```
num3 DOUBLE PRECISION;
```

```
BEGIN
```

```
null;
```

```
END;
```

Output

```
1 DECLARE
2     num1 INTEGER;
3     num2 REAL;
4     num3 DOUBLE PRECISION;
5 BEGIN
6     null;
7 END;
8
```

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

```

1 DECLARE
2     SUBTYPE name IS char(20);
3     SUBTYPE message IS varchar2(100);
4     salutation name;
5     greetings message;
6 BEGIN
7     salutation := 'Reader ';
8     greetings := 'Welcome to the World of PL/SQL';
9     dbms_output.put_line('Hello ' || salutation || greetings);
10 END;
11

```

```

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

```

Variable Declaration in PL/SQL

Program

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;

Output

```
Statement processed.  
Value of c: 30  
Value of f: 23.3333333333333333333333333333333333333333333333333
```

BEGIN

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

Output

```
1 DECLARE
2   -- Global variables
3   num1 number := 95;
4   num2 number := 85;
5 BEGIN
6   dbms_output.put_line('Outer Variable num1: ' || num1);
7   dbms_output.put_line('Outer Variable num2: ' || num2);
8   DECLARE
9     -- Local variables
10    num1 number := 195;
11    num2 number := 185;
12  BEGIN
13    dbms_output.put_line('Inner Variable num1: ' || num1);
14    dbms_output.put_line('Inner Variable num2: ' || num2);
15  END;
16 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)
```

);

```
1 CREATE TABLE CUSTOMERS(  
2     ID INT NOT NULL,  
3     NAME VARCHAR (20) NOT NULL,  
4     AGE INT NOT NULL,  
5     ADDRESS CHAR (25),  
6     SALARY DECIMAL (18, 2),  
7     PRIMARY KEY (ID)  
8 );  
9
```

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 );
```



```

1  INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
2  VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
3
4  INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
5  VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );
6
7  INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
8  VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );
9
10 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
11 VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );
12
13 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
14 VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );
15
16 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
17 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.

Program

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

```
1 DECLARE
2     c_id customers.id%type := 1;
3     c_name customers.name%type;
4     c_addr customers.address%type;
5     c_sal customers.salary%type;
6 BEGIN
7     SELECT name, address, salary INTO c_name, c_addr, c_sal
8     FROM customers
9     WHERE id = c_id;
10    dbms_output.put_line
11    ('Customer ' || c_name || ' from ' || c_addr || ' earns ' || c_sal);
12 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

```
1  PI CONSTANT NUMBER := 3.141592654;  
2  DECLARE  
3      -- constant declaration  
4      pi constant number := 3.141592654;  
5      -- other declarations  
6      radius number(5,2);  
7      dia number(5,2);  
8      circumference number(7, 2);  
9      area number (10, 2);  
10 BEGIN  
11     -- processing  
12     radius := 9.5;  
13     dia := radius * 2;  
14     circumference := 2.0 * pi * radius;  
15     area := pi * radius * radius;  
16     -- output |  
17     dbms_output.put_line('Radius: ' || radius);  
18     dbms_output.put_line('Diameter: ' || dia);  
19     dbms_output.put_line('Circumference: ' || circumference);  
20     dbms_output.put_line('Area: ' || area);  
21 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

1	DECLARE	
2	message varchar2(30):= 'That's tutorialspoint.com!';	
3	BEGIN	
4	dbms_output.put_line(message);	
5	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

```
1 BEGIN
2   dbms_output.put_line( 10 + 5);
3   dbms_output.put_line( 10 - 5);
4   dbms_output.put_line( 10 * 5);
5   dbms_output.put_line( 10 / 5);
6   dbms_output.put_line( 10 ** 5);
7 END;
```

Statement processed.

15
5
50
2
100000

PL/SQL - Relational Operators

Program

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

```

1 DECLARE
2   a number (2) := 21;
3   b number (2) := 10;
4 BEGIN
5   IF (a = b) then
6     dbms_output.put_line('Line 1 - a is equal to b');
7   ELSE
8     dbms_output.put_line('Line 1 - a is not equal to b');
9   END IF;
10  IF (a < b) then
11    dbms_output.put_line('Line 2 - a is less than b');
12  ELSE
13    dbms_output.put_line('Line 2 - a is not less than b');
14  END IF;
15
16  IF ( a > b ) THEN
17    dbms_output.put_line('Line 3 - a is greater than b');
18  ELSE
19    dbms_output.put_line('Line 3 - a is not greater than b');
20  END IF;
21  -- Lets change value of a and b
22  a := 5;
23  b := 20;
24  IF ( a <= b ) THEN
25    dbms_output.put_line('Line 4 - a is either equal or less than b');
26  END IF;
27  IF ( b >= a ) THEN
28    dbms_output.put_line('Line 5 - b is either equal or greater than a');
29  END IF;
30  IF ( a <> b ) THEN

```

Statement processed.

15

5

50

2

100000

PL/SQL - Comparison Operators

Program

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

<pre> 1 DECLARE 2 PROCEDURE compare (value varchar2, pattern varchar2) is 3 BEGIN 4 IF value LIKE pattern THEN 5 dbms_output.put_line ('True'); 6 ELSE 7 dbms_output.put_line ('False'); 8 END IF; 9 END; 10 BEGIN 11 compare('Zara Ali', 'Z%A_i'); 12 compare('Nuha Ali', 'Z%A_i'); 13 END; </pre>	<pre> Statement processed. True False </pre>
--	--

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

```

1 DECLARE
2     a boolean := true;
3     b boolean := false;
4 BEGIN
5     IF (a AND b) THEN
6         dbms_output.put_line('Line 1 - Condition is true');
7     END IF;
8     IF (a OR b) THEN
9         dbms_output.put_line('Line 2 - Condition is true');
10    END IF;
11    IF (NOT a) THEN
12        dbms_output.put_line('Line 3 - a is not true');
13    ELSE
14        dbms_output.put_line('Line 3 - a is true');
15    END IF;
16    IF (NOT b) THEN
17        dbms_output.put_line('Line 4 - b is not true');
18    ELSE
19        dbms_output.put_line('Line 4 - b is true');
20    END IF;
21 END;

```

```

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

```

PL/SQL Operator Precedence

Program

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

```
1 DECLARE
2   a number(2) := 20;
3   b number(2) := 10;
4   c number(2) := 15;
5   d number(2) := 5;
6   e number(2);
7 BEGIN
8   e := (a + b) * c / d;      -- ( 30 * 15 ) / 5
9   dbms_output.put_line('Value of (a + b) * c / d is : ' || e);
10  e := ((a + b) * c) / d;    -- (30 * 15) / 5
11  dbms_output.put_line('Value of ((a + b) * c) / d is : ' || e);
12  e := (a + b) * (c / d);    -- (30) * (15/5)
13  dbms_output.put_line('Value of (a + b) * (c / d) is : ' || e);
14  e := a + (b * c) / d;      -- 20 + (150/5)
15  dbms_output.put_line('Value of a + (b * c) / d is : ' || e);
16 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

```
1 DECLARE
2   a number(2) := 10;
3 BEGIN
4   a:= 10;
5   -- check the boolean condition using if statement
6   IF( a < 20 ) THEN
7     -- if condition is true then print the following
8     dbms_output.put_line('a is less than 20 ' );
9   END IF;
10  dbms_output.put_line('value of a is : ' || a);
11 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

<pre>1 DECLARE 2 a number(3) := 100; 3 BEGIN 4 -- check the boolean condition using if statement 5 IF(a < 20) THEN 6 -- if condition is true then print the following 7 dbms_output.put_line('a is less than 20 '); 8 ELSE 9 dbms_output.put_line('a is not less than 20 '); 10 END IF; 11 dbms_output.put_line('value of a is : ' a); 12 END;</pre>	
--	--

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

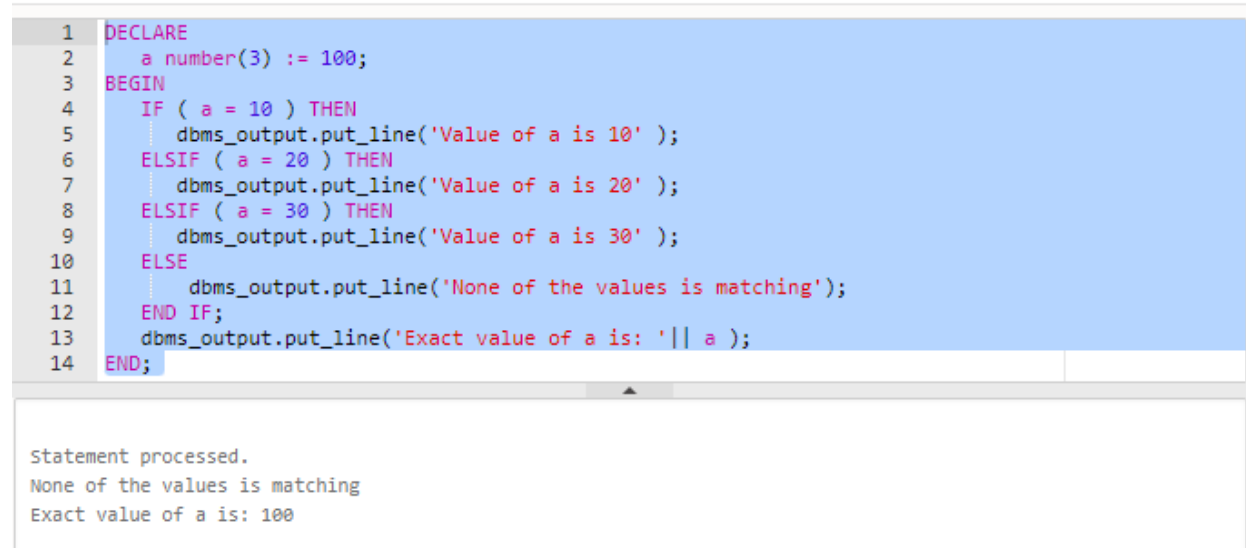
dbms_output.put_line('None of the values is matching');

END IF;

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

```
END;
```

Output



The screenshot shows a SQL IDE with a PL/SQL program in the editor and its output in the console. The program is a simple IF-THEN-ELSE statement that checks the value of a variable 'a'. Since 'a' is 100, none of the conditions (a = 10, 20, 30) are met, so it prints 'None of the values is matching' and then 'Exact value of a is: 100'.

```
1 DECLARE
2   a number(3) := 100;
3 BEGIN
4   IF ( a = 10 ) THEN
5     dbms_output.put_line('Value of a is 10' );
6   ELSIF ( a = 20 ) THEN
7     dbms_output.put_line('Value of a is 20' );
8   ELSIF ( a = 30 ) THEN
9     dbms_output.put_line('Value of a is 30' );
10  ELSE
11    dbms_output.put_line('None of the values is matching');
12  END IF;
13  dbms_output.put_line('Exact value of a is: ' || a );
14 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

```

1 DECLARE
2   grade char(1) := 'A';
3 BEGIN
4   CASE grade
5     when 'A' then dbms_output.put_line('Excellent');
6     when 'B' then dbms_output.put_line('Very good');
7     when 'C' then dbms_output.put_line('Well done');
8     when 'D' then dbms_output.put_line('You passed');
9     when 'F' then dbms_output.put_line('Better try again');
10    else dbms_output.put_line('No such grade');
11  END CASE;
12 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

```

1 DECLARE
2   grade char(1) := 'B';
3 BEGIN
4   case
5     when grade = 'A' then dbms_output.put_line('Excellent');
6     when grade = 'B' then dbms_output.put_line('Very good');
7     when grade = 'C' then dbms_output.put_line('Well done');
8     when grade = 'D' then dbms_output.put_line('You passed');
9     when grade = 'F' then dbms_output.put_line('Better try again');
10    else dbms_output.put_line('No such grade');
11  end case;
12 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


```

1 DECLARE
2     a number(3) := 100;
3     b number(3) := 200;
4 BEGIN
5     -- check the boolean condition
6     IF( a = 100 ) THEN
7         -- if condition is true then check the following
8         IF( b = 200 ) THEN
9             -- if condition is true then print the following
10            dbms_output.put_line('Value of a is 100 and b is 200' );
11        END IF;
12    END IF;
13    dbms_output.put_line('Exact value of a is : ' || a );
14    dbms_output.put_line('Exact value of b is : ' || b );
15 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

```

1 DECLARE
2   x number := 10;
3 BEGIN
4   LOOP
5     dbms_output.put_line(x);
6     x := x + 10;
7     exit WHEN x > 50;
8   END LOOP;
9   -- after exit, control resumes here
10  dbms_output.put_line('After Exit x is: ' || x);
11 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

```
1 DECLARE
2     a number(2) := 10;
3 BEGIN
4     WHILE a < 20 LOOP
5         dbms_output.put_line('value of a: ' || a);
6         a := a + 1;
7     END LOOP;
8 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

```

1 DECLARE
2   a number(2);
3 BEGIN
4   FOR a in 10 .. 20 LOOP
5     dbms_output.put_line('value of a: ' || a);
6   END LOOP;
7 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

Program

```

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

```
1 DECLARE
2   i number(3);
3   j number(3);
4 BEGIN
5   i := 2;
6   LOOP
7     j:= 2;
8     LOOP
9       exit WHEN ((mod(i, j) = 0) or (j = i));
10      j := j +1;
11    END LOOP;
12    IF (j = i ) THEN
13      dbms_output.put_line(i || ' is prime');
14    END IF;
15    i := i + 1;
16    exit WHEN i = 50;
17  END LOOP;
18 END;
19 /
```

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);

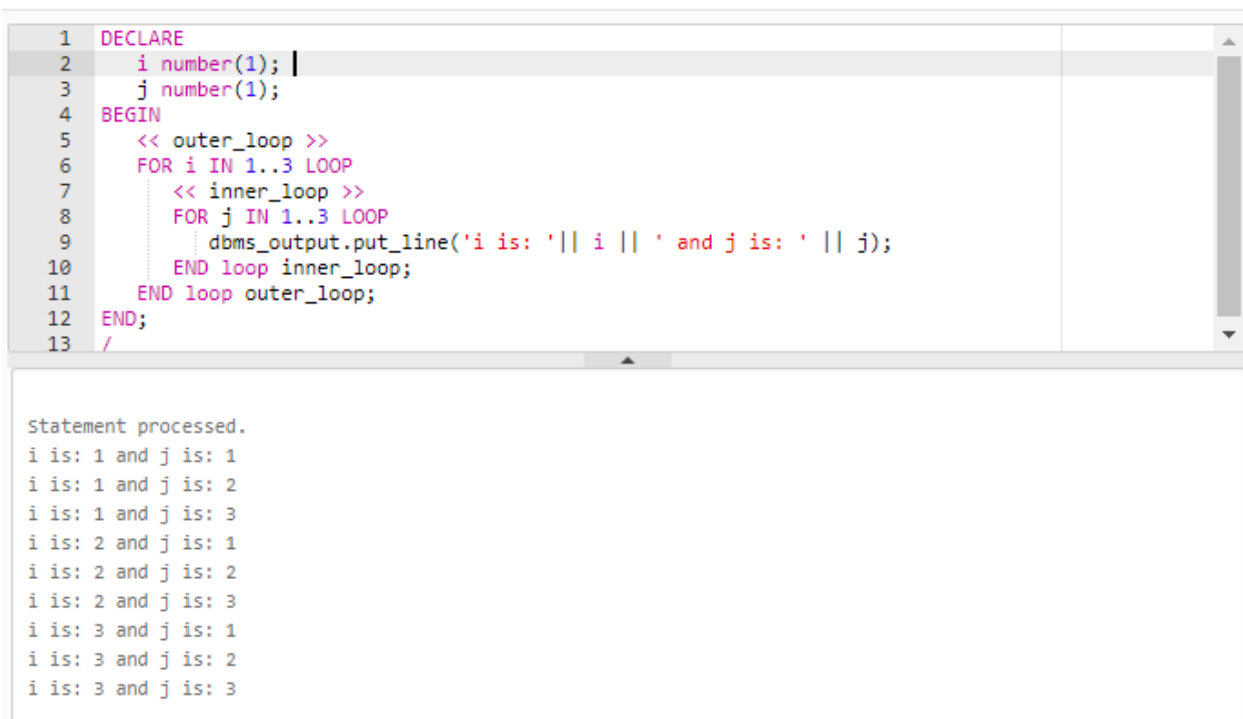
BEGIN
```

```

<< 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



The screenshot shows a SQL Developer window with a PL/SQL script in the top pane and its execution output in the bottom pane. The script is a nested loop that prints the values of variables i and j. The output shows the results of the execution, confirming that the nested loop executed correctly for all combinations of i and j from 1 to 3.

```

1 DECLARE
2     i number(1); |
3     j number(1);
4 BEGIN
5     << outer_loop >>
6     FOR i IN 1..3 LOOP
7         << inner_loop >>
8         FOR j IN 1..3 LOOP
9             dbms_output.put_line('i is: ' || i || ' and j is: ' || j);
10        END loop inner_loop;
11    END loop outer_loop;
12 END;
13 /

```

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;
/
```

Output

<pre>1 DECLARE 2 a number(2) := 10; 3 BEGIN 4 -- while loop execution 5 WHILE a < 20 LOOP 6 dbms_output.put_line ('value of a: ' a); 7 a := a + 1; 8 IF a > 15 THEN 9 -- terminate the loop using the exit statement 10 EXIT; 11 END IF; 12 END LOOP; 13 END; 14 /</pre>	
--	--


```
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


```

1 DECLARE
2     a number(2) := 10;
3 BEGIN
4     -- while loop execution
5     WHILE a < 20 LOOP
6         dbms_output.put_line ('value of a: ' || a);
7         a := a + 1;
8         IF a = 15 THEN
9             -- skip the loop using the CONTINUE statement
10            a := a + 1;
11            CONTINUE;
12        END IF;
13    END LOOP;
14 END; |
15 /
16

```

```

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

```
1 DECLARE
2   a number(2) := 10;
3 BEGIN
4   <<loopstart>>
5   -- while loop execution
6   WHILE a < 20 LOOP
7     dbms_output.put_line ('value of a: ' || a);
8     a := a + 1;
9     IF a = 15 THEN
10      a := a + 1;
11      GOTO loopstart;
12    END IF;
13  END LOOP;
14 END;
15 /
16
```

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';

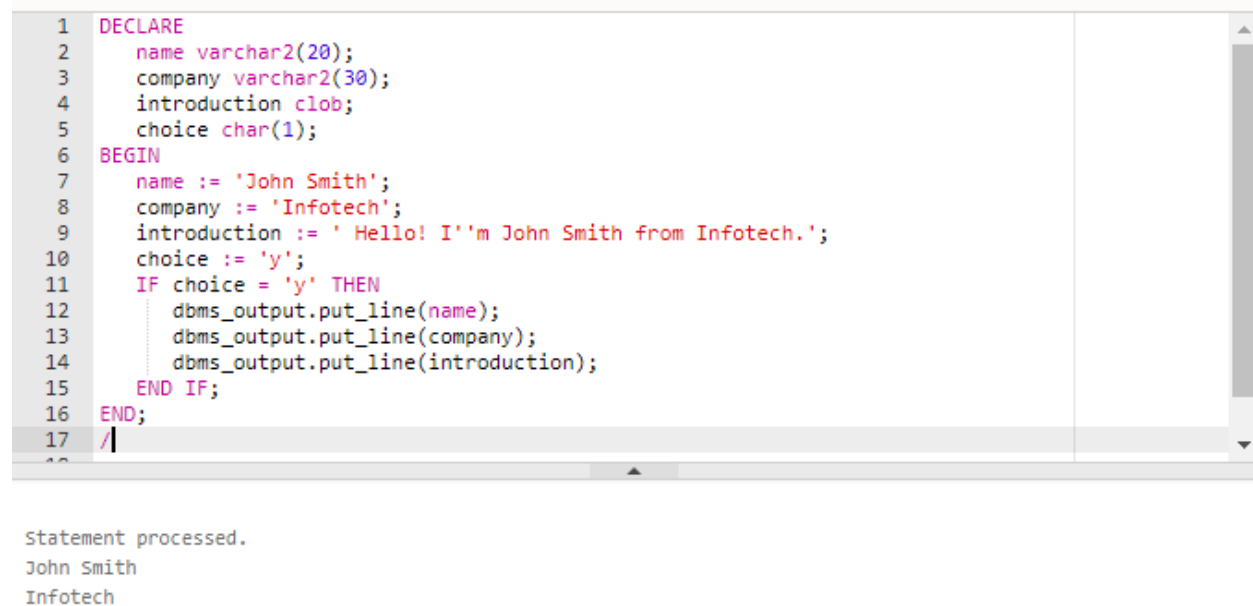
company := 'Infotech';

```

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



```

1 DECLARE
2     name varchar2(20);
3     company varchar2(30);
4     introduction clob;
5     choice char(1);
6 BEGIN
7     name := 'John Smith';
8     company := 'Infotech';
9     introduction := ' Hello! I'm John Smith from Infotech.';
10    choice := 'y';
11    IF choice = 'y' THEN
12        dbms_output.put_line(name);
13        dbms_output.put_line(company);
14        dbms_output.put_line(introduction);
15    END IF;
16 END;
17 /

```

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

```
1 DECLARE
2   greetings varchar2(11) := 'hello world';
3 BEGIN
4   dbms_output.put_line(UPPER(greetings));
5
6   dbms_output.put_line(LOWER(greetings));
7
8   dbms_output.put_line(INITCAP(greetings));
9
10  /* retrieve the first character in the string */
11  dbms_output.put_line ( SUBSTR (greetings, 1, 1));
12
13  /* retrieve the last character in the string */
14  dbms_output.put_line ( SUBSTR (greetings, -1, 1));
15
16  /* retrieve five characters,
17   starting from the seventh position. */
18  dbms_output.put_line ( SUBSTR (greetings, 7, 5));
19
20  /* retrieve the remainder of the string,
21   starting from the second position. */
22  dbms_output.put_line ( SUBSTR (greetings, 2));
23
24  /* find the location of the first "e" */
25  dbms_output.put_line ( INSTR (greetings, 'e'));
26 END;
27 /
28
```

Statement processed.
HELLO WORLD
hello world
Hello World
h

PL/SQL – Arrays

Program

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

```

1 DECLARE
2     type namesarray IS VARRAY(5) OF VARCHAR2(10);
3     type grades IS VARRAY(5) OF INTEGER;
4     names namesarray;
5     marks grades;
6     total integer;
7 BEGIN
8     names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
9     marks:= grades(98, 97, 78, 87, 92);
10    total := names.count;
11    dbms_output.put_line('Total '|| total || ' Students');
12    FOR i in 1 .. total LOOP
13        dbms_output.put_line('Student: ' || names(i) || '
14        Marks: ' || marks(i));
15    END LOOP;
16 END;
17 /

```

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

```

1 DECLARE
2     a number;
3     b number;
4     c number;
5 PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
6 BEGIN
7     IF x < y THEN
8         z:= x;
9     ELSE
10        z:= y;
11    END IF;
12 END;
13 BEGIN
14     a:= 23;
15     b:= 45;
16     findMin(a, b, c);
17     dbms_output.put_line(' Minimum of (23, 45) : ' || c);
18 END;
19 /
20
21

```

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

PL/SQL – Functions

Program

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;
/

```

Output

```

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;
17 BEGIN
18     a:= 23;
19     b:= 45;
20     c := findMax(a, b);
21     dbms_output.put_line(' Maximum of (23,45): ' || c);
22 END;
23 /

```

```

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

```

1 DECLARE
2     num number;
3     factorial number;
4
5 FUNCTION fact(x number)
6 RETURN number
7 IS
8     f number;
9 BEGIN
10    IF x=0 THEN
11        f := 1;
12    ELSE
13        f := x * fact(x-1);
14    END IF;
15 RETURN f;
16 END;
17 |
18 BEGIN
19     num:= 6;
20     factorial := fact(num);
21     dbms_output.put_line(' Factorial ' || num || ' is ' || factorial);
22 END;
23 /

```

Statement processed.
Factorial 6 is 720

PL/SQL – Cursors

Program

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

```
1 DECLARE
2   c_id customers.id%type;
3   c_name customers.name%type;
4   c_addr customers.address%type;
5   CURSOR c_customers is
6     SELECT id, name, address FROM customers;
7 BEGIN
8   OPEN c_customers;
9   LOOP
10    FETCH c_customers into c_id, c_name, c_addr;
11    EXIT WHEN c_customers%notfound;
12    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
13  END LOOP;
14  CLOSE c_customers;
15 END;
16 /
```

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

```
1 DECLARE  
2     customer_rec customers%rowtype;  
3 BEGIN  
4     SELECT * into customer_rec  
5     FROM customers  
6     WHERE id = 5;  
7     dbms_output.put_line('Customer ID: ' || customer_rec.id);  
8     dbms_output.put_line('Customer Name: ' || customer_rec.name);  
9     dbms_output.put_line('Customer Address: ' || customer_rec.address);  
10    dbms_output.put_line('Customer Salary: ' || customer_rec.salary);  
11 END;  
12 /
```

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

```
1  DECLARE
2      c_id customers.id%type := 8;
3      c_name customerS.Name%type;
4      c_addr customers.address%type;
5  BEGIN
6      SELECT name, address INTO c_name, c_addr
7      FROM customers
8      WHERE id = c_id;
9      DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
10     DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr); _
11
12  EXCEPTION
13      WHEN no_data_found THEN
14          dbms_output.put_line('No such customer!');
15      WHEN others THEN
16          dbms_output.put_line('Error!');
17  END;
18  /
19
```

```
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

<pre> 1 CREATE OR REPLACE TRIGGER display_salary_changes 2 BEFORE DELETE OR INSERT OR UPDATE ON customers 3 FOR EACH ROW 4 WHEN (NEW.ID > 0) 5 DECLARE 6 sal_diff number; 7 BEGIN 8 sal_diff := :NEW.salary - :OLD.salary; 9 dbms_output.put_line('Old salary: ' :OLD.salary); 10 dbms_output.put_line('New salary: ' :NEW.salary); 11 dbms_output.put_line('Salary difference: ' sal_diff); 12 END; 13 / </pre>	<pre> Trigger created. </pre>
--	-------------------------------

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

1	CREATE OR REPLACE PACKAGE c_package AS	
2	-- Adds a customer	
3	PROCEDURE addCustomer(c_id customers.id%type,	
4	c_name customers.Name%type,	
5	c_age customers.age%type,	
6	c_addr customers.address%type,	
7	c_sal customers.salary%type);	
8		
9	-- Removes a customer	
10	PROCEDURE delCustomer(c_id customers.id%TYPE); _	
11	--Lists all customers	
12	PROCEDURE listCustomer;	
13		
14	END c_package;	
15	/	
16		

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

```

1 DECLARE
2     TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
3     salary_list salary;
4     name VARCHAR2(20);
5 BEGIN
6     -- adding elements to the table
7     salary_list('Rajnish') := 62000;
8     salary_list('Minakshi') := 75000;
9     salary_list('Martin') := 100000;
10    salary_list('James') := 78000;
11
12    -- printing the table
13    name := salary_list.FIRST; _
14    WHILE name IS NOT null LOOP
15        dbms_output.put_line
16        ('Salary of ' || name || ' is ' || TO_CHAR(salary_list(name)));
17        name := salary_list.NEXT(name);
18    END LOOP;
19 END;
20 /

```

```

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

```

1
2 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
3 VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );
4
5 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
6 VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );
7
8 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
9 VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );
10
11 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
12 VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );
13
14 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
15 VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );
16
17 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
18 VALUES (6, 'Komal', 22, 'MP', 4500.00 );
19
20 COMMIT;
21 PDATE CUSTOMERS
22 SET SALARY = SALARY + 1000;
23 ROLLBACK TO sav1;
24
25 UPDATE CUSTOMERS
26 SET SALARY = SALARY + 1000
27 WHERE ID = 7;
28 UPDATE CUSTOMERS
29 SET SALARY = SALARY + 1000
30 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

```

1 SELECT SYSDATE FROM DUAL;
2 SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY HH:MI:SS') FROM DUAL;
3 SELECT ADD_MONTHS(SYSDATE, 5) FROM DUAL;
4 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

```

1 DECLARE
2     lines dbms_output.chararr;
3     num_lines number;
4 BEGIN
5     -- enable the buffer with default size 20000
6     dbms_output.enable;
7
8     dbms_output.put_line('Hello Reader!');
9     dbms_output.put_line('Hope you have enjoyed the tutorials!');
10    dbms_output.put_line('Have a great time exploring pl/sql!');
11
12    num_lines := 3;
13
14    dbms_output.get_lines(lines, num_lines);
15
16    FOR i IN 1..num_lines LOOP
17        dbms_output.put_line(lines(i));
18    END LOOP;
19 END;
20 /

```

```

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

```

PL/SQL - Object Oriented

Program

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

```
1 CREATE OR REPLACE TYPE rectangle AS OBJECT
2 (length number,
3  width number,
4  member function enlarge( inc number) return rectangle,
5  member procedure display,
6  map member function measure return number
7 );
8 /
```

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

```
1 CREATE OR REPLACE TYPE BODY rectangle AS
2     MEMBER FUNCTION enlarge(inc number) return rectangle IS
3     BEGIN
4         return rectangle(self.length + inc, self.width + inc);
5     END enlarge;
6     MEMBER PROCEDURE display IS
7     BEGIN
8         dbms_output.put_line('Length: ' || length);
9         dbms_output.put_line('Width: ' || width);
10    END display;
11    MAP MEMBER FUNCTION measure return number IS
12    BEGIN
13        return (sqrt(length*length + width*width));
14    END measure;
15 END;
16 / _
```

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;

/

Output

```
1 DECLARE
2     r1 rectangle;
3     r2 rectangle;
4     r3 rectangle;
5     inc_factor number := 5;
6 BEGIN
7     r1 := rectangle(3, 4);
8     r2 := rectangle(5, 7);
9     r3 := r1.enlarge(inc_factor);
10    r3.display;
11    IF (r1 > r2) THEN -- calling measure function
12        r1.display;
13    ELSE
14        r2.display; _
15    END IF;
16 END;
17 /
```

Statement processed.

Length: 8

Width: 9

Length: 5

Width: 7