DECLARE

monthly\_salary NUMBER(6);

number\_of\_days\_worked NUMBER(2);

pay\_per\_day NUMBER(6,2);

-- the following is the executable part, from BEGIN to END

BEGIN

monthly\_salary := 22222;

number\_of\_days\_worked := 21;

pay\_per\_day := monthly\_salary/number\_of\_days\_worked;

-- the following displays output from the PL/SQL block

DBMS\_OUTPUT.PUT\_LINE('The pay per day is ' || pay\_per\_day);

-- the following is an optional exception part that handles errors

EXCEPTION

WHEN ZERO\_DIVIDE THEN

pay\_per\_day := 0;

END;

2) DECLARE

answer VARCHAR2(20); -- declare a variable

BEGIN

-- assign a value to a variable

answer := 'Maybe';

-- use PUT\_LINE to display data from the PL/SQL block

DBMS\_OUTPUT.PUT\_LINE( 'The answer is: ' || answer );

END;

3)

DECLARE -- declare and assiging variables

wages NUMBER(6,2);

hours\_worked NUMBER := 40;

hourly\_salary NUMBER := 22.50;

bonus NUMBER := 150;

country VARCHAR2(128);

counter NUMBER := 0;

done BOOLEAN := FALSE;

valid\_id BOOLEAN;

BEGIN

wages := (hours\_worked \* hourly\_salary) + bonus; -- compute wages

country := 'France'; -- assign a string literal

country := UPPER('Canada'); -- assign an uppercase string literal

done := (counter > 100); -- assign a BOOLEAN, in this case FALSE

valid\_id := TRUE; -- assign a BOOLEAN

dbms\_output.put\_line('country is'||country);

END;

DECLARE -- declare and assign variables

number1 PLS\_INTEGER := 32000; -- numeric literal

number2 NUMBER(8,3);

BEGIN

number2 := 3.125346e3; -- numeric literal

number2 := -8300.00; -- numeric literal

number2 := -14; -- numeric literal

END;

/

Assigning Values to Variables Using PL/SQL SELECT INTO

DECLARE -- declare and assign values

bonus\_rate CONSTANT NUMBER(2,3) := 0.05;

bonus NUMBER(8,2);

emp\_id NUMBER(6) := 120; -- assign a test value for employee ID

BEGIN

-- retreive a salary from the employees table, then calculate the bonus and

-- assign the value to the bonus variable

SELECT salary \* bonus\_rate INTO bonus FROM employees

WHERE employee\_id = emp\_id;

-- display the employee\_id, bonus amount, and bonus rate

DBMS\_OUTPUT.PUT\_LINE ( 'Employee: ' || TO\_CHAR(emp\_id)

|| ' Bonus: ' || TO\_CHAR(bonus) || ' Bonus Rate: ' || TO\_CHAR(bonus\_rate));

END;

/

DECLARE -- declare variables using %TYPE attribute

empid employees.employee\_id%TYPE; -- employee\_id datatype is NUMBER(6)

emplname employees.last\_name%TYPE; -- last\_name datatype is VARCHAR2(25)

BEGIN

empid := 100301; -- this is OK because it fits in NUMBER(6)

-- empid := 3018907; -- this is too large and will cause an overflow

emplname := 'Patel'; -- this is OK because it fits in VARCHAR2(25)

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || empid); -- display data

DBMS\_OUTPUT.PUT\_LINE('Employee name: ' || emplname); -- display data

END;

/

Using %ROWTYPE with a PL/SQL Record

DECLARE -- declare variables

-- declare record variable that represents a row fetched from the employees table

emp\_rec employees%ROWTYPE; -- declare variable with %ROWTYPE attribute

BEGIN

SELECT \* INTO emp\_rec FROM EMPLOYEES WHERE employee\_id = 120; -- retrieve record

DBMS\_OUTPUT.PUT\_LINE('Employee name: ' || emp\_rec.first\_name || ' '

|| emp\_rec.last\_name); -- display

END;

/

DECLARE

emp\_rec employee%ROWTYPE;

my\_empno employee.dno%TYPE := 100;

CURSOR c1 IS

SELECT department\_id, department\_name, location\_id FROM department;

dept\_rec c1%ROWTYPE;

BEGIN

SELECT \* INTO emp\_rec FROM employees WHERE dno = my\_empno;

IF (emp\_rec.department\_id = 20) AND (emp\_rec.salary > 2000) THEN

NULL;

END IF;

END;

Using a Simple IF-THEN Statement in PL/SQL

DECLARE

sal NUMBER(8,2);

bonus NUMBER(6,2);

hiredate DATE;

empid NUMBER(6) := 128; -- use employee 120 for testing

BEGIN

-- retrieve the salary and the date that employee was hired, the date is checked

-- to calculate the amount of the bonus for the employee

SELECT salary, hire\_date INTO sal, hiredate FROM employees

WHERE employee\_id = empid;

IF hiredate > TO\_DATE('01-JAN-00') THEN

bonus := sal/20;

DBMS\_OUTPUT.PUT\_LINE('Bonus for employee: ' || empid || ' is: ' || bonus );

END IF;

END;

/

Using the IF-THEN-ELSEIF Statement in PL/SQL

DECLARE

bonus NUMBER(6,2);

empid NUMBER(6) := 120;

hiredate DATE;

BEGIN

-- retrieve the date that employee was hired, the date is checked

-- to determine the amount of the bonus for the employee

SELECT hire\_date INTO hiredate FROM employees WHERE employee\_id = empid;

IF hiredate > TO\_DATE('01-JAN-98') THEN

bonus := 500;

ELSIF hiredate > TO\_DATE('01-JAN-96') THEN

bonus := 1000;

ELSE

bonus := 1500;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Bonus for employee: ' || empid || ' is: ' || bonus );

END;

/

Using the FOR-LOOP in PL/SQL

BEGIN

-- use a FOR loop to process a series of numbers

FOR loop\_counter IN 1..10 LOOP

DBMS\_OUTPUT.PUT\_LINE('Number: ' || TO\_CHAR(loop\_counter)

|| ' Square: ' || TO\_CHAR(loop\_counter\*\*2));

END LOOP;

END;

/

Using WHILE-LOOP for Control in PL/SQL

DECLARE -- declare variables

i NUMBER := 1; -- loop counter, initialize to one

i\_cubed NUMBER;

BEGIN

-- use WHILE LOOP to process data

WHILE i <= 10 LOOP

i\_cubed := i\*\*3;

DBMS\_OUTPUT.PUT\_LINE('Number: ' || TO\_CHAR(i)

|| ' Cube: ' || TO\_CHAR(i\_cubed));

i := i + 1;

END LOOP;

END;

/

Using the EXIT-WHEN Statement in PL/SQL

DECLARE -- declare and assign values to variables

total NUMBER(9) := 0;

counter NUMBER(6) := 0;

BEGIN

LOOP

counter := counter + 1; -- increment counter variable

total := total + counter \* counter; -- compute total

-- exit loop when condition is true

EXIT WHEN total > 25000; -- LOOP until condition is met

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Counter: ' || TO\_CHAR(counter)

|| ' Total: ' || TO\_CHAR(total)); -- display results

END;

/