



P
L
/
S
Q
L



Oracle11g: PL/SQL Programming

Chapter 2

Basic PL/SQL Block Structures



Chapter Objectives

- After completing this lesson, you should be able to understand:
 - Programming fundamentals
 - PL/SQL blocks
 - How to define and declare variables
 - How to initialize and manage variable values
 - The NOT NULL and CONSTANT variable options



Chapter Objectives (continued)

- After completing this lesson, you should be able to understand (continued):
 - How to perform calculations with variables
 - The use of SQL single-row functions in PL/SQL statements
 - Decision structures: IF-THEN and CASE
 - Looping actions: basic, FOR and WHILE
 - CONTINUE statements
 - Nested Statements



Program Logic Flow

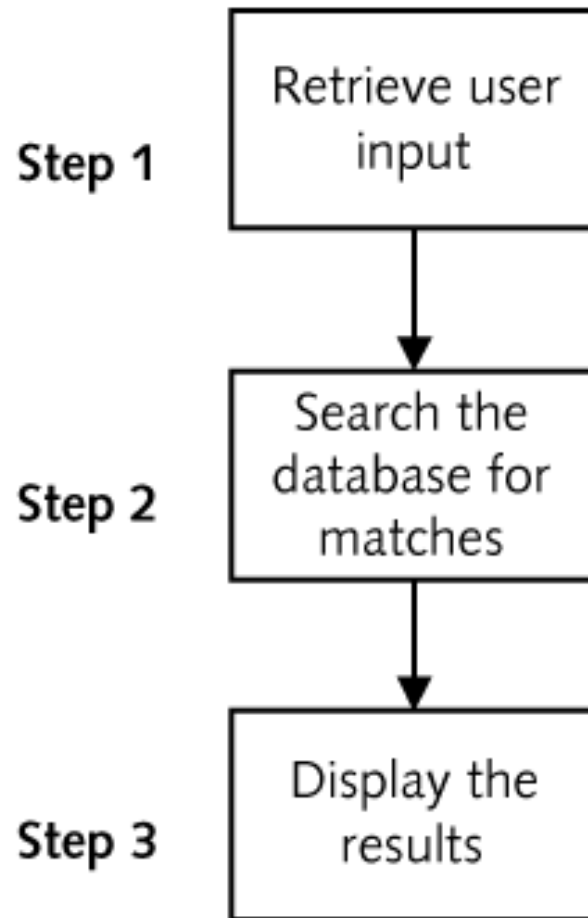
P
L
/
S
Q
L

- Identify sequence of actions needed prior to coding
- Use a flowchart to visually represent the sequence of actions



Flowcharting - Search for Coffee Products

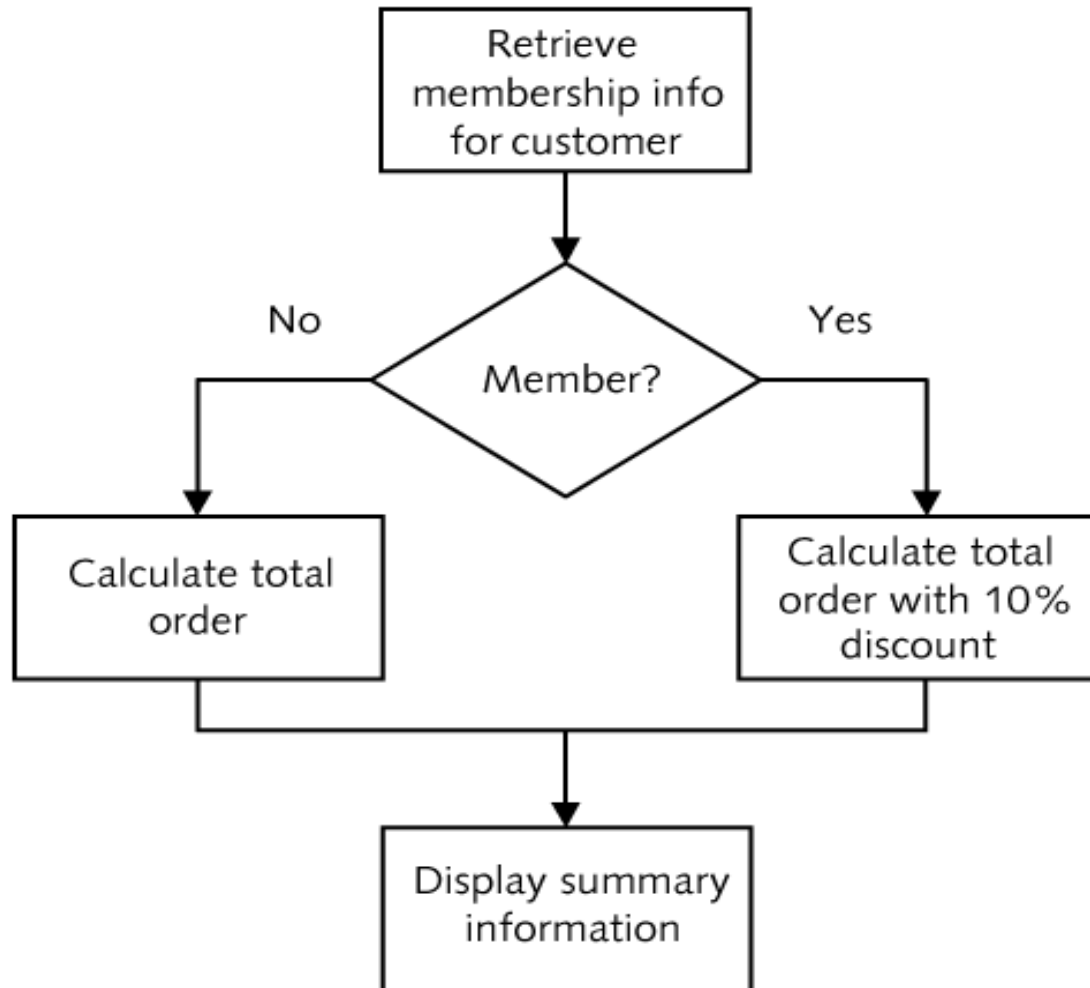
P
L
/
S
Q
L





Decision Structures

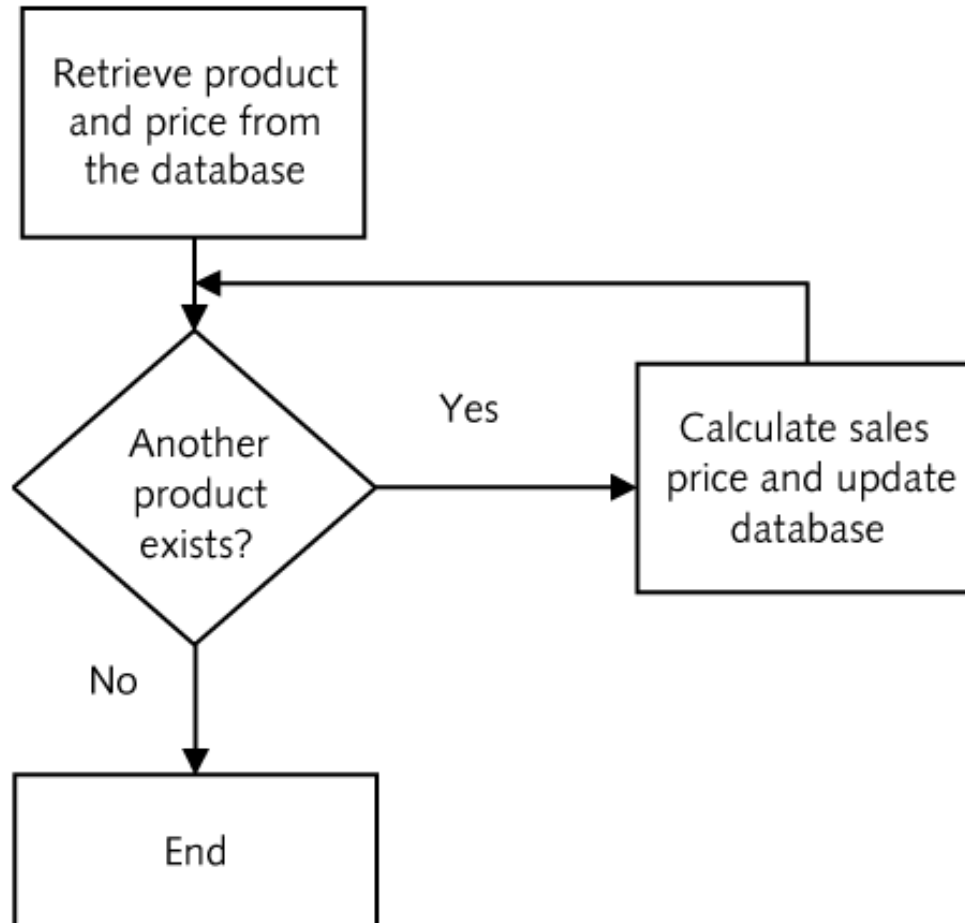
P
L
/
S
Q
L





Looping Structures

P
L
/
S
Q
L





PL/SQL Block Questions


P
L
/
S
Q
L

- What is a block?
- What are the different segments of a block?
- How does data get into a block?
- How are different data types handled?




Brewbean's Challenge


P
L
/
S
Q
L



Brewbean's Coffee Shop


[Departments](#)


Click **here** to continue shopping


[Basket](#)


Item Code	Name	Options	Qty	Price	Total
7	Columbia	1 lb., Whole Bean	<input type="text" value="1"/>	\$10.80	\$10.80
9	Ethiopia	1 lb., Whole Bean	<input type="text" value="1"/>	\$10.00	\$10.00

Subtotal: \$20.80

[Check Out](#)

[Search](#)

[Account](#)

[Order Status](#)

[Remove](#) [Remove](#)



PL/SQL Block Structure

P
L
/
S
Q
L

- **DECLARE** – create variables, cursors, and types
- **BEGIN** – SQL, logic, loops, assignment statements
- **EXCEPTION** – error handling
- **END** – close the block



Variable Names

P
L
/
S
Q
L

- Begin with alpha character
- Up to 30 characters
- Can contain upper and lowercase letters, numbers, _ , \$, #



Scalar Variable Data Types

- **Character** – CHAR(n)
VARCHAR2(n)
- **Numeric** – NUMBER(p,s)
- **Date** – DATE
- **Boolean** – BOOLEAN

Note: Only holds a single value



Example Scalar Declarations

P
L
/
S
Q
L

```
DECLARE
    lv_ord_date DATE;
    lv_last_txt VARCHAR2(25);
    lv_qty_num NUMBER(2);
    lv_shipflag_bln BOOLEAN;
BEGIN
    ----- PL/SQL executable statements -----
END;
```

Note: Minimum requirements are variable name and data type



Test Variables

P
L
/
S
Q
L

Run Script button

```
1 DECLARE
2     lv_ord_date DATE;
3     lv_last_txt VARCHAR2(25);
4     lv_qty_num NUMBER(2);
5     lv_shipflag_bln BOOLEAN;
6     lv_bln_txt VARCHAR2(5);
7 BEGIN
8     lv_ord_date := '12-JUL-2012';
9     lv_last_txt := 'Brown';
10    lv_qty_num := 3;
11    lv_shipflag_bln := TRUE;
12    DBMS_OUTPUT.PUT_LINE(lv_ord_date);
13    DBMS_OUTPUT.PUT_LINE(lv_last_txt);
14    DBMS_OUTPUT.PUT_LINE(lv_qty_num);
15    IF lv_shipflag_bln THEN
16        lv_bln_txt := 'OK';
17    END IF;
18    DBMS_OUTPUT.PUT_LINE(lv_bln_txt);
19 END;
```



Variable Initialization

P
L
/
S
Q
L

- Set a variable value when the variable is created

```
DECLARE
```

```
    lv_ord_date DATE := SYSDATE;
```

```
    lv_last_txt VARCHAR2(25) := 'Unknown';
```

```
    lv_qty_num NUMBER(2) := 0;
```

```
    lv_shipflag_bln BOOLEAN := 'FALSE';
```

```
BEGIN
```

```
    ---- PL/SQL executable statements ----
```

```
END;
```



Test Variable Initialization

P
L
/
S
Q
L

The screenshot displays the Oracle SQL Developer environment. The main window, titled 'XE_plbook', shows a PL/SQL script in the 'Worksheet' tab. The script declares four variables: `lv_ord_date` (DATE), `lv_last_txt` (VARCHAR2(25)), `lv_qty_num` (NUMBER(2)), and `lv_shipflag_bln` (BOOLEAN). It then uses `DBMS_OUTPUT.PUT_LINE` to output the values of these variables. The script is executed, and the 'Script Output' and 'Dbms Output' tabs are visible at the bottom. The 'Script Output' tab shows 'Task completed in 0.015 seconds' and 'anonymous block completed'. The 'Dbms Output' tab shows the output: '15-JAN-12', 'Unknown', and '0'.

```
1 DECLARE
2     lv_ord_date DATE := SYSDATE;
3     lv_last_txt VARCHAR2(25) := 'Unknown';
4     lv_qty_num NUMBER(2) := 0;
5     lv_shipflag_bln BOOLEAN := FALSE;
6 BEGIN
7     DBMS_OUTPUT.PUT_LINE(lv_ord_date);
8     DBMS_OUTPUT.PUT_LINE(lv_last_txt);
9     DBMS_OUTPUT.PUT_LINE(lv_qty_num);
10 END;
```

Script Output x

Task completed in 0.015 seconds

anonymous block completed

Dbms Output x

Buffer Size: 20000

15-JAN-12
Unknown
0



Variable Declaration Options

P
L
/
S
Q
L

- NOT NULL – the variable must always contain a value
- CONSTANT – the variable value can not be changed in the block

```
DECLARE
```

```
    lv_shipcntry_txt VARCHAR2(15) NOT NULL := 'US';
```

```
    lv_taxrate_num CONSTANT NUMBER(2,2) := .06;
```

```
BEGIN
```

```
    ---- PL/SQL executable statements ----
```

```
END;
```



Calculations with Scalar Variables

P
L
/
S
Q
L

multiplication

```
DECLARE
  lv_taxrate_num CONSTANT NUMBER(2,2) := .06;
  lv_total_num NUMBER(6,2) := 50;
  lv_taxamt_num NUMBER(4,2);
BEGIN
  lv_taxamt_num := lv_total_num * lv_taxrate_num;
  DBMS_OUTPUT.PUT_LINE(lv_taxamt_num);
END;
/
```



Using SQL Functions

P
L
/
S
Q
L

- SQL functions such as MONTHS_BETWEEN can be used within PL/SQL statements

The screenshot shows the XE_plbook SQL Developer window. The main editor displays a PL/SQL block with the following code:

```
1 DECLARE
2   lv_first_date DATE := '20-OCT-2012';
3   lv_second_date DATE := '20-SEP-2010';
4   lv_months_num NUMBER(3);
5 BEGIN
6   lv_months_num := MONTHS_BETWEEN(lv_first_date,lv_second_date);
7   DBMS_OUTPUT.PUT_LINE(lv_months_num);
8 END;
```

An arrow points from the text 'SQL functions such as MONTHS_BETWEEN can be used within PL/SQL statements' to the `MONTHS_BETWEEN` function call in line 6 of the code.

Below the code editor, the 'Script Output' pane shows the message 'Task completed in 0.016 seconds' and 'anonymous block completed'. The 'Dbms Output' pane shows the output '25'.



Decision Structures

- Control which statements in a PL/SQL block will execute
- Enables conditions to be tested to determine the flow of statement execution
- Most programming languages provide IF and CASE statements to enable conditional processing



Decision Structures (continued)

- IF Statements
 - Simple IF
 - IF/THEN/ELSE
 - IF/THEN/ELSIF/ELSE
- CASE Statements
 - Basic CASE statement
 - Searched CASE statement
 - CASE expression



Basic IF Statement

PL/SQL

The screenshot shows the Oracle SQL Developer environment. The main window is titled 'XE_plbook' and contains a 'Worksheet' tab. The script being edited is as follows:

```
1 DECLARE
2   lv_state_txt CHAR(2) := 'VA';
3   lv_sub_num NUMBER(5,2) := 100;
4   lv_tax_num NUMBER(4,2) := 0;
5 BEGIN
6   IF lv_state_txt = 'VA' THEN
7     lv_tax_num := lv_sub_num * .06;
8   END IF;
9   DBMS_OUTPUT.PUT_LINE(lv_tax_num);
10 END;
```

Below the script editor, there are two output panes. The 'Script Output' pane shows the message 'anonymous block completed'. The 'Dbms Output' pane shows the value '6', which is the result of the calculation performed in the IF statement.



IF/THEN/ELSE

P
L
/
S
Q
L

The screenshot shows the Oracle SQL Developer interface. The main window is titled "XE_plbook" and contains a "Query Builder" pane. The script being executed is as follows:

```
1 DECLARE
2   lv_state_txt CHAR(2) := 'NC';
3   lv_sub_num NUMBER(5,2) := 100;
4   lv_tax_num NUMBER(4,2) := 0;
5 BEGIN
6   IF lv_state_txt = 'VA' THEN
7     lv_tax_num := lv_sub_num * .06;
8   ELSE
9     lv_tax_num := lv_sub_num * .04;
10  END IF;
11  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
12 END;
```

Below the script pane, there are two output panes. The "Script Output" pane shows the message "anonymous block completed". The "Dbms Output" pane shows the value "4", which is the result of the calculation 100×0.04 for the state 'NC'.



IF/THEN/ELSIF/ELSE

P
L
/
S
Q
L

The screenshot displays the Oracle SQL Developer environment. The main window, titled 'XE_plbook', shows a PL/SQL script in the 'Worksheet' tab. The script defines three local variables: `lv_state_txt` (CHAR(2)), `lv_sub_num` (NUMBER(5,2)), and `lv_tax_num` (NUMBER(4,2)). It then enters a `BEGIN` block with an `IF` statement. The `IF` statement checks the value of `lv_state_txt`. If it is 'VA', it calculates `lv_tax_num` as `lv_sub_num * .06`. If it is 'ME', it calculates `lv_tax_num` as `lv_sub_num * .05`. If it is 'NY', it calculates `lv_tax_num` as `lv_sub_num * .07`. If none of these conditions are met, it goes to the `ELSE` clause and calculates `lv_tax_num` as `lv_sub_num * .04`. Finally, it uses `DBMS_OUTPUT.PUT_LINE(lv_tax_num);` to display the result and ends with `END;`. The 'Script Output' pane below shows the message 'anonymous block completed' and the execution time 'Task completed in 0.016 seconds'. The 'Dbms Output' pane shows the result '5'. The 'XE_plbook' tab is visible at the bottom.

```
1 DECLARE
2   lv_state_txt CHAR(2) := 'ME';
3   lv_sub_num NUMBER(5,2) := 100;
4   lv_tax_num NUMBER(4,2) := 0;
5 BEGIN
6   IF lv_state_txt = 'VA' THEN
7     lv_tax_num := lv_sub_num * .06;
8   ELSIF lv_state_txt = 'ME' THEN
9     lv_tax_num := lv_sub_num * .05;
10  ELSIF lv_state_txt = 'NY' THEN
11    lv_tax_num := lv_sub_num * .07;
12  ELSE
13    lv_tax_num := lv_sub_num * .04;
14  END IF;
15  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
16 END;
```

Script Output x

Task completed in 0.016 seconds

anonymous block completed

Dbms Output x

Buffer Size: 20000

5

XE_plbook x



Nested IF

P
L
/
S
Q
L

```
IF lv_type_txt = 'E' THEN
  IF lv_price_num > 85 THEN    <-- Inner or nested IF begins
    lv_disc_num = .20;
  ELSIF lv_price_num > 45 THEN
    lv_disc_num = .15;
  ELSE
    lv_disc_num = .10;
  END IF;                    <-- Inner or nested IF ends
ELSIF lv_type_txt = 'C' THEN
  lv_disc_num = .05;
END IF;
```



Logical Operators within IF

- Logical operators (AND, OR) enable multiple conditions to be checked

```
IF lv_state_txt = 'VA' OR lv_state_txt = 'PA' THEN
    lv_tax_num := lv_sub_num * .06;
ELSE
    lv_tax_num := lv_sub_num * .04;
END IF;
```



Basic CASE Statement

P
L
/
S
Q
L

The screenshot displays the Oracle SQL Developer environment. The main window, titled 'XE_plbook', contains a 'Worksheet' tab with a SQL script. The script defines three local variables: 'lv_state_txt' (CHAR(2)), 'lv_sub_num' (NUMBER(5,2)), and 'lv_tax_num' (NUMBER(4,2)). It then uses a CASE statement to calculate 'lv_tax_num' based on the value of 'lv_state_txt'. The script is executed, and the 'Script Output' window shows 'Task completed in 0.016 seconds' and 'anonymous block completed'. The 'Dbms Output' window shows the result '5'.

```
1 DECLARE
2   lv_state_txt CHAR(2) := 'ME';
3   lv_sub_num NUMBER(5,2) := 100;
4   lv_tax_num NUMBER(4,2) := 0;
5 BEGIN
6   CASE lv_state_txt
7     WHEN 'VA' THEN lv_tax_num := lv_sub_num * .06;
8     WHEN 'ME' THEN lv_tax_num := lv_sub_num * .05;
9     WHEN 'NY' THEN lv_tax_num := lv_sub_num * .07;
10    ELSE lv_tax_num := lv_sub_num * .04;
11  END CASE;
12  DBMS_OUTPUT.PUT_LINE(lv_tax_num);
13 END;
```

Script Output x

Task completed in 0.016 seconds

anonymous block completed

Dbms Output x

Buffer Size: 20000

5



Searched CASE

P
L
/
S
Q
L

The screenshot shows the Oracle SQL Developer interface. The main window is titled 'XE_plbook' and contains a 'Worksheet' tab. The script being executed is as follows:

```
1 DECLARE
2   lv_state_txt CHAR(2) := 'VA';
3   lv_zip_txt CHAR(5) := '23321';
4   lv_sub_num NUMBER(5,2) := 100;
5   lv_tax_num NUMBER(4,2) := 0;
6 BEGIN
7   CASE
8     WHEN lv_zip_txt = '23321' THEN
9       lv_tax_num := lv_sub_num * .02;
10    WHEN lv_state_txt = 'VA' THEN
11      lv_tax_num := lv_sub_num * .06;
12    ELSE
13      lv_tax_num := lv_sub_num * .04;
14    END CASE;
15   DBMS_OUTPUT.PUT_LINE(lv_tax_num);
16 END;
```

Below the script editor, there are two output windows. The 'Script Output' window shows the message 'anonymous block completed' and indicates that the task was completed in 0.016 seconds. The 'Dbms Output' window shows the output of the script, which is the number '2'. The 'XE_plbook' window is also visible at the bottom.



CASE Expression

PL/SQL

The screenshot shows the Oracle SQL Developer interface. The main window is titled 'XE_plbook' and contains a 'Query Builder' tab. The script being executed is as follows:

```
1 DECLARE
2   lv_state_txt CHAR(2) := 'ME';
3   lv_sub_num NUMBER(5,2) := 100;
4   lv_tax_num NUMBER(4,2) := 0;
5 BEGIN
6   lv_tax_num := CASE lv_state_txt
7     WHEN 'VA' THEN lv_sub_num * .06
8     WHEN 'ME' THEN lv_sub_num * .05
9     WHEN 'NY' THEN lv_sub_num * .07
10    ELSE lv_sub_num * .04
11  END;
12   DBMS_OUTPUT.PUT_LINE(lv_tax_num);
13 END;
```

Below the script editor, the 'Script Output' window shows the message 'anonymous block completed'. The 'Dbms Output' window shows the result '5'. The 'XE_plbook' window is also visible at the bottom.



Looping

- Enables a statement or set of statements to be executed more than once
- A loop must provide instructions of when to end the looping, or an 'infinite' loop will be produced



Basic LOOP

P
L
/
S
Q
L

The screenshot shows the Oracle SQL Developer interface with a window titled 'XE_plbook'. The 'Worksheet' tab is active, displaying a PL/SQL script. The script defines a variable 'lv_cnt_num' of type NUMBER(2) and sets it to 1. It then enters a 'LOOP' block where it prints the value of 'lv_cnt_num' using 'DBMS_OUTPUT.PUT_LINE', checks for an exit condition 'lv_cnt_num >= 5', and increments the counter by 1. The loop ends with 'END LOOP;' and the entire block is terminated with 'END;'. Below the script, the 'Script Output' pane shows the message 'anonymous block completed'. The 'Dbms Output' pane is also visible, showing a list of output lines numbered 1 through 5, corresponding to the five iterations of the loop.

```
1 DECLARE
2     lv_cnt_num NUMBER(2) := 1;
3 BEGIN
4     LOOP
5         DBMS_OUTPUT.PUT_LINE(lv_cnt_num);
6         EXIT WHEN lv_cnt_num >= 5;
7         lv_cnt_num := lv_cnt_num + 1;
8     END LOOP;
9 END;
```

Script Output x

Task completed in 0 seconds

anonymous block completed

Dbms Output x

Buffer Size: 20000

1
2
3
4
5



WHILE Loop

P
L
/
S
Q
L

The screenshot shows the Oracle SQL Developer interface. The main window is titled 'XE_plbook' and contains a 'Worksheet' tab. The script in the worksheet is as follows:

```
1 DECLARE
2     lv_cnt_num NUMBER(2) := 1;
3 BEGIN
4     WHILE lv_cnt_num <= 5 LOOP
5         DBMS_OUTPUT.PUT_LINE(lv_cnt_num);
6         lv_cnt_num := lv_cnt_num + 1;
7     END LOOP;
8 END;
```

Below the script, the 'Script Output' pane shows the message 'Task completed in 0.016 seconds' and 'anonymous block completed'. The 'Dbms Output' pane shows the output of the loop, which is the numbers 1 through 5, each on a new line.



FOR Loop

P
L
/
S
Q
L

The screenshot displays the Oracle SQL Developer environment. The main window, titled 'XE_plbook', contains a 'Query Builder' tab with a PL/SQL script. The script is as follows:

```
1 BEGIN
2   FOR i IN 1..5 LOOP
3     DBMS_OUTPUT.PUT_LINE(i);
4   END LOOP;
5 END;
```

Below the script editor, the 'Script Output' pane shows the message 'Task completed in 0.031 seconds' and 'anonymous block completed'. The 'Dbms Output' pane is also visible, showing a list of numbers 1 through 5, which are the output of the loop. The 'XE_plbook' tab is active at the bottom.



CONTINUE Statement

P
L
/
S
Q
L

The screenshot displays the Oracle SQL Developer environment. The main window, titled 'XE_plbook', shows a PL/SQL script in the 'Query Builder' tab. The script is as follows:

```
1 DECLARE
2   lv_cnt_num NUMBER(3) := 0;
3 BEGIN
4   FOR i IN 1..25 LOOP
5     CONTINUE WHEN MOD(i,5) <> 0;
6     DBMS_OUTPUT.PUT_LINE('Loop i value: ' || i);
7     lv_cnt_num := lv_cnt_num + 1;
8   END LOOP;
9   DBMS_OUTPUT.PUT_LINE('Final execution count: ' || lv_cnt_num);
10 END;
```

Below the script editor, the 'Script Output' window shows the message 'anonymous block completed' and 'Task completed in 0.046 seconds'. The 'Dbms Output' window, with a buffer size of 20000, displays the following output:

```
Loop i value: 5
Loop i value: 10
Loop i value: 15
Loop i value: 20
Loop i value: 25
Final execution count: 5
```



Nested Loops

P
L
/
S
Q
L

The screenshot shows the Oracle SQL Developer interface. The top toolbar includes icons for running, saving, and other database operations. The main window is titled 'XE_plbook' and contains a 'Worksheet' tab with the following PL/SQL code:

```
1 BEGIN
2   FOR oi IN 1..3 LOOP
3     DBMS_OUTPUT.PUT_LINE('Outer Loop');
4     FOR ii IN 1..2 LOOP
5       DBMS_OUTPUT.PUT_LINE('Inner Loop');
6     END LOOP;
7   END LOOP;
8 END;
```

Below the code editor, the 'Script Output' pane shows the message 'Task completed in 0.015 seconds' and 'anonymous block completed'. The 'Dbms Output' pane shows the output of the script, which is a sequence of 'Outer Loop' and 'Inner Loop' messages, demonstrating the execution of the nested loops.

Outer Loop
Inner Loop
Inner Loop
Outer Loop
Inner Loop
Inner Loop
Outer Loop
Inner Loop
Inner Loop



Summary

- A flowchart assists in laying out processing logic
- A PL/SQL block contains a DECLARE, BEGIN, EXCEPTION, and END sections
- Variables to hold values are declared
- Scalar variables hold a single data value
- Scalar variables can hold string values, numbers, dates, and Boolean values
- DBMS_OUTPUT.PUT_LINE is used to display values



Summary (continued)

- IF statement structure is IF/THEN/ELSIF/ELSE
- CASE statements provide decision processing similar to IF statements
- Looping structures include: basic, WHILE, and FOR
- Host or bind variables can be used to interact with the application environment