Chapter 2 Solutions

Review Questions

1. b
2. c
3. b
4. c
5. b
6. d
7. a
8. a.
9. b and c
10. b
11. Variables are structures used to hold data values during the processing of a block.
12. A basic loop uses an `EXIT WHEN` statement to stop. A `WHILE` loop uses a condition in the beginning of the loop to determine whether it should continue. A `FOR` loop uses a range in the beginning of the loop to determine how many times the loop iterates.
13. `IF/THEN` and `CASE` statements
14. A flowchart is a tool developers can use to map out the logic sequence needed to prepare for coding a block.
15. A `CONSTANT` option doesn't allow changing an initialized variable's value during block execution.

Advanced Review Questions

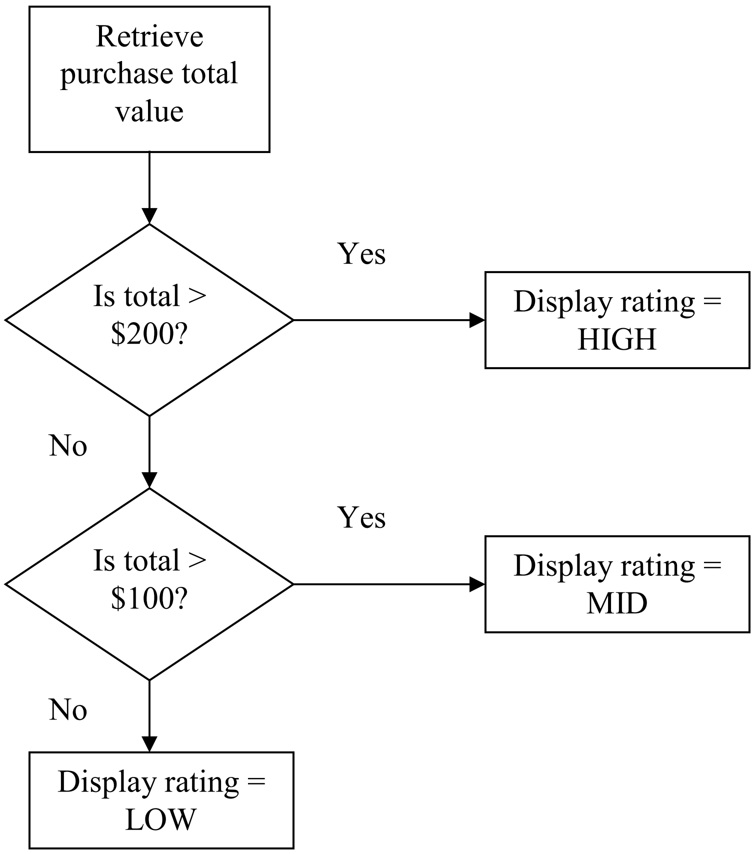
1. b
2. a
3. d
4. d
5. c

Hands-On Assignments Part I

Assignment 2-1

```
DECLARE
  lv_test_date DATE := '10-DEC-2012';
  lv_test_num NUMBER(3) := 10;
  lv_test_txt VARCHAR2(10);
BEGIN
  lv_test_txt :=  '???????';
  DBMS_OUTPUT.PUT_LINE(lv_test_date);
  DBMS_OUTPUT.PUT_LINE(lv_test_num);
  DBMS_OUTPUT.PUT_LINE(lv_test_txt);
END;
```

Assignment 2-2



Assignment 2-3

```
DECLARE
  lv_total_num NUMBER(6,2) := 150;
BEGIN
  IF lv_total_num > 200 THEN
    DBMS_OUTPUT.PUT_LINE('HIGH');
  ELSIF lv_total_num > 100 THEN
    DBMS_OUTPUT.PUT_LINE('MID');
  ELSE
    DBMS_OUTPUT.PUT_LINE('LOW');
  END IF;
END;
```

Assignment 2-4

```
DECLARE
  lv_total_num NUMBER(6,2) := 150;
```

```
BEGIN
  CASE
    WHEN lv_total_num  > 200 THEN
      DBMS_OUTPUT.PUT_LINE('HIGH');
    WHEN lv_total_num > 100 THEN
      DBMS_OUTPUT.PUT_LINE('MID');
    ELSE
      DBMS_OUTPUT.PUT_LINE('LOW');
    END CASE;
END;
```

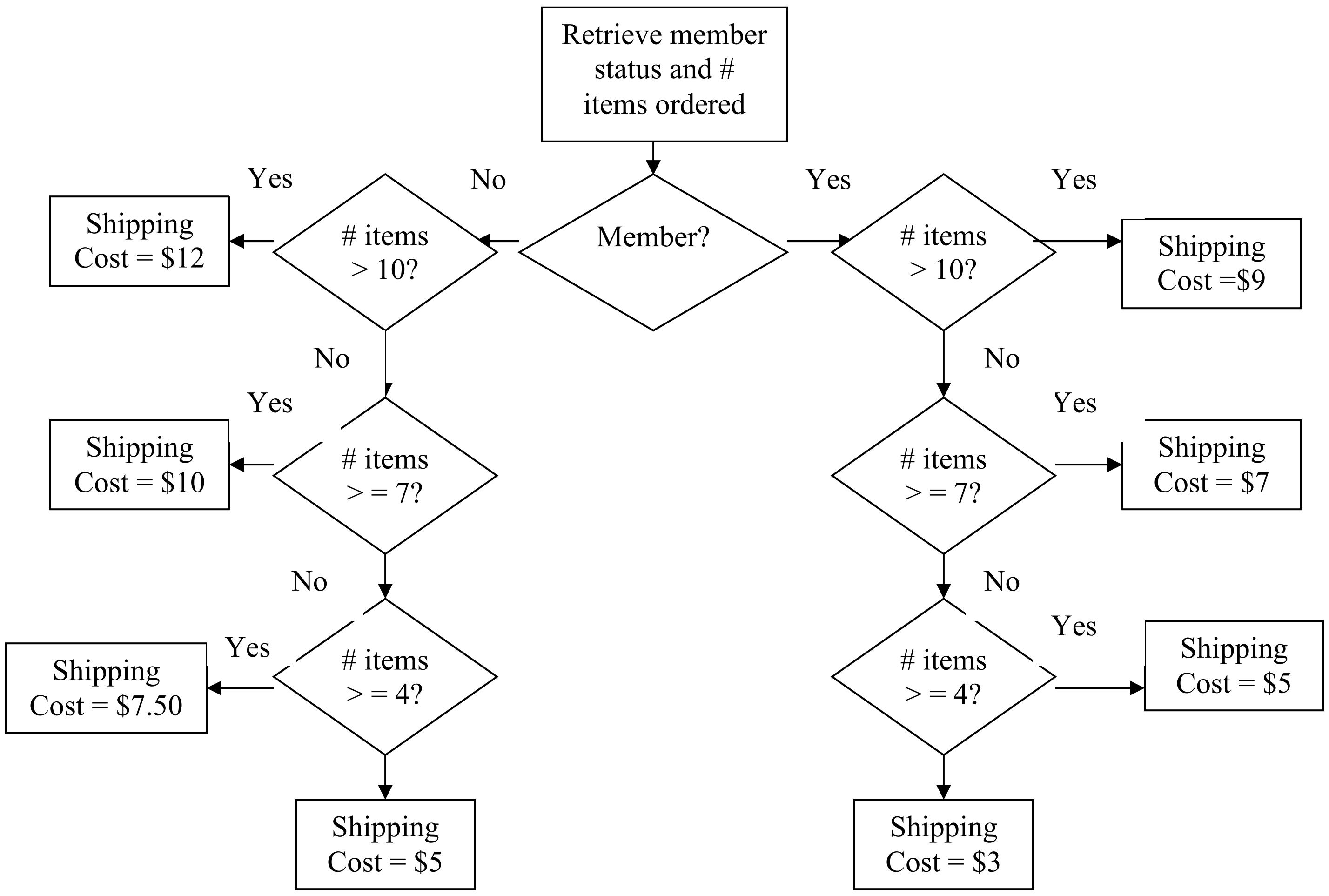
Assignment 2-5

```
DECLARE
  lv_bal_num NUMBER(8,2) := 150.50;
  lv_pay_num NUMBER(8,2) := 95.00;
  lv_due_bln BOOLEAN;
BEGIN
  IF (lv_bal_num - lv_pay_num) > 0 THEN
    lv_due_bln := TRUE;
    DBMS_OUTPUT.PUT_LINE('Balance Due');
   ELSE
    lv_due_bln := FALSE;
    DBMS_OUTPUT.PUT_LINE('Account Fully Paid');
  END IF;
END;
```

Assignment 2-6

```
DECLARE
  lv_total_num NUMBER(6,2) := 200;
  lv_price_num NUMBER(5,2) := 32;
  lv_spent_num NUMBER(6,2) := 0;
  lv_qty_num NUMBER(6) := 0;
BEGIN
  WHILE (lv_spent_num + lv_price_num) < lv_total_num LOOP
    lv_spent_num := lv_spent_num + lv_price_num;
    lv_qty_num := lv_qty_num + 1;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Total Spent = ' || lv_spent_num);
  DBMS_OUTPUT.PUT_LINE('# purchase = ' || lv_qty_num);
END;
```

Assignment 2-7



Assignment 2-8

```
DECLARE
  lv_mem_txt CHAR(1) := 'Y';
  lv_qty_num NUMBER(3) := 8;
  lv_ship_num NUMBER(6,2);
BEGIN
  IF lv_mem_txt =  'Y' THEN
    IF lv_qty_num > 10 THEN
      lv_ship_num := 9;
    ELSIF lv_qty_num >= 7 THEN
      lv_ship_num := 7;
    ELSIF lv_qty_num >= 4 THEN
      lv_ship_num := 5;
    ELSE
      lv_ship_num := 3;
    END IF;
  ELSE
    IF lv_qty_num > 10 THEN
      lv_ship_num := 12;
    ELSIF lv_qty_num >= 7 THEN
      lv_ship_num := 10;
    ELSIF lv_qty_num >= 4 THEN
      lv_ship_num := 7.50;
    ELSE
      lv_ship_num := 5;
    END IF;
  END IF;
  DBMS_OUTPUT.PUT_LINE(lv_ship_num);
END;
```

Hands-On Assignments Part II

Assignment 2-9

```
DECLARE
  lv_start_date DATE := '01-OCT-2012';
  lv_payamt_num NUMBER(8,2) := 20;
  lv_paymths_num NUMBER(8,2) := 24;
  lv_bal_num NUMBER(8,2) := 0;
  lv_duedate_date DATE;
  lv_duedate_txt VARCHAR2(25);
BEGIN
  lv_bal_num := lv_payamt_num * lv_paymths_num;
  lv_duedate_date := lv_start_date;
   FOR i IN 1..lv_paymths_num LOOP
     lv_bal_num := lv_bal_num - lv_payamt_num;
     lv_duedate_txt := TO_CHAR(lv_duedate_date,'mm/dd/yyyy');
     DBMS_OUTPUT.PUT_LINE('Pay #: ' || i || ' Due: ' || lv_duedate_txt
     || ' Amt: ' || TO_CHAR(lv_payamt_num,'$999.99')
     || ' Bal: ' || TO_CHAR(lv_bal_num,'$9,999.99'));
     lv_duedate_date :=  ADD_MONTHS(lv_duedate_date,1);
  END LOOP;
END;
```

Assignment 2-10

```
DECLARE
  lv_start_date DATE := '01-OCT-2012';
  lv_payamt_num NUMBER(8,2) := 20;
  lv_paymths_num NUMBER(8,2) := 24;
  lv_bal_num NUMBER(8,2) := 0;
  lv_duedate_date DATE;
  lv_duedate_txt VARCHAR2(25);
  lv_cnt_num NUMBER(2) := 1;
BEGIN
  lv_bal_num := lv_payamt_num * lv_paymths_num;
  lv_duedate_date := lv_start_date;
  LOOP
    lv_bal_num := lv_bal_num - lv_payamt_num;
    lv_duedate_txt := TO_CHAR(lv_duedate_date,'mm/dd/yyyy');
    DBMS_OUTPUT.PUT_LINE('Pay #: ' || lv_cnt_num || ' Due: ' ||
lv_duedate_txt
    || ' Amt: ' || TO_CHAR(lv_payamt_num,'$999.99')
    || ' Bal: ' || TO_CHAR(lv_bal_num,'$9,999.99'));
    lv_duedate_date :=  ADD_MONTHS(lv_duedate_date,1);
      EXIT WHEN (lv_cnt_num = lv_paymths_num);
      lv_cnt_num := lv_cnt_num + 1;
  END LOOP;
END;
```

Assignment 2-11

```
DECLARE
  lv_start_date DATE := '01-OCT-2012';
  lv_payamt_num NUMBER(8,2) := 20;
  lv_paymths_num NUMBER(8,2) := 24;
  lv_paid_num NUMBER(8,2) := 0;
  lv_duedate_date DATE;
  lv_duedate_txt VARCHAR2(25);
  lv_cnt_num NUMBER(2) := 1;
BEGIN
  lv_duedate_date := lv_start_date;
  WHILE lv_cnt_num <= lv_paymths_num LOOP
    lv_paid_num := lv_paid_num + lv_payamt_num;
    lv_duedate_txt := TO_CHAR(lv_duedate_date,'mm/dd/yyyy');
    DBMS_OUTPUT.PUT_LINE('Pay #: ' || lv_cnt_num || ' Due: ' ||
       lv_duedate_txt || ' Amt: ' || TO_CHAR(lv_payamt_num,'$999.99')
       || ' Total Paid: ' || TO_CHAR(lv_paid_num,'$9,999.99'));
    lv_duedate_date :=  ADD_MONTHS(lv_duedate_date,1);
    lv_cnt_num := lv_cnt_num + 1;
  END LOOP;
END;
```

Assignment 2-12

```
DECLARE
  lv_paycode_num NUMBER(1) := 1;
  lv_payamt_num NUMBER(6,2) := 100;
  lv_match_num NUMBER(8,2) := 0;
BEGIN
  lv_match_num := CASE lv_paycode_num
    WHEN 0 THEN lv_payamt_num * .25
    WHEN 1 THEN lv_payamt_num * .5
    WHEN 2 THEN lv_payamt_num * 1
    ELSE 0
  END;
  DBMS_OUTPUT.PUT_LINE(lv_match_num);
END;
```

Assignment 2-13

```
DECLARE
  lv_type_txt CHAR(1) := 'B';
  lv_amt_num NUMBER(8,2) := 600;
  lv_match_num NUMBER(8,2) := 0;
BEGIN
```

```
  IF lv_type_txt = 'I' THEN
    IF lv_amt_num >= 500 THEN
        lv_match_num := lv_amt_num * .20;
      ELSIF lv_amt_num >= 250 THEN
        lv_match_num := lv_amt_num * .30;
      ELSIF lv_amt_num >= 100 THEN
        lv_match_num := lv_amt_num * .50;
      ELSE
        lv_match_num := 0;
     END IF;
  ELSIF  lv_type_txt = 'B' THEN
      IF lv_amt_num >= 1000 THEN
        lv_match_num := lv_amt_num * .05;
      ELSIF lv_amt_num >= 500 THEN
        lv_match_num := lv_amt_num * .10;
      ELSIF lv_amt_num >= 100 THEN
        lv_match_num := lv_amt_num * .20;
      ELSE
        lv_match_num := 0;
    END IF;
  ELSIF lv_type_txt = 'G' THEN
    IF lv_amt_num >= 100 THEN
         lv_match_num := lv_amt_num * .05;
      ELSE
        lv_match_num := 0;
    END IF;
  END IF;
  DBMS_OUTPUT.PUT_LINE(lv_match_num);
END;
```

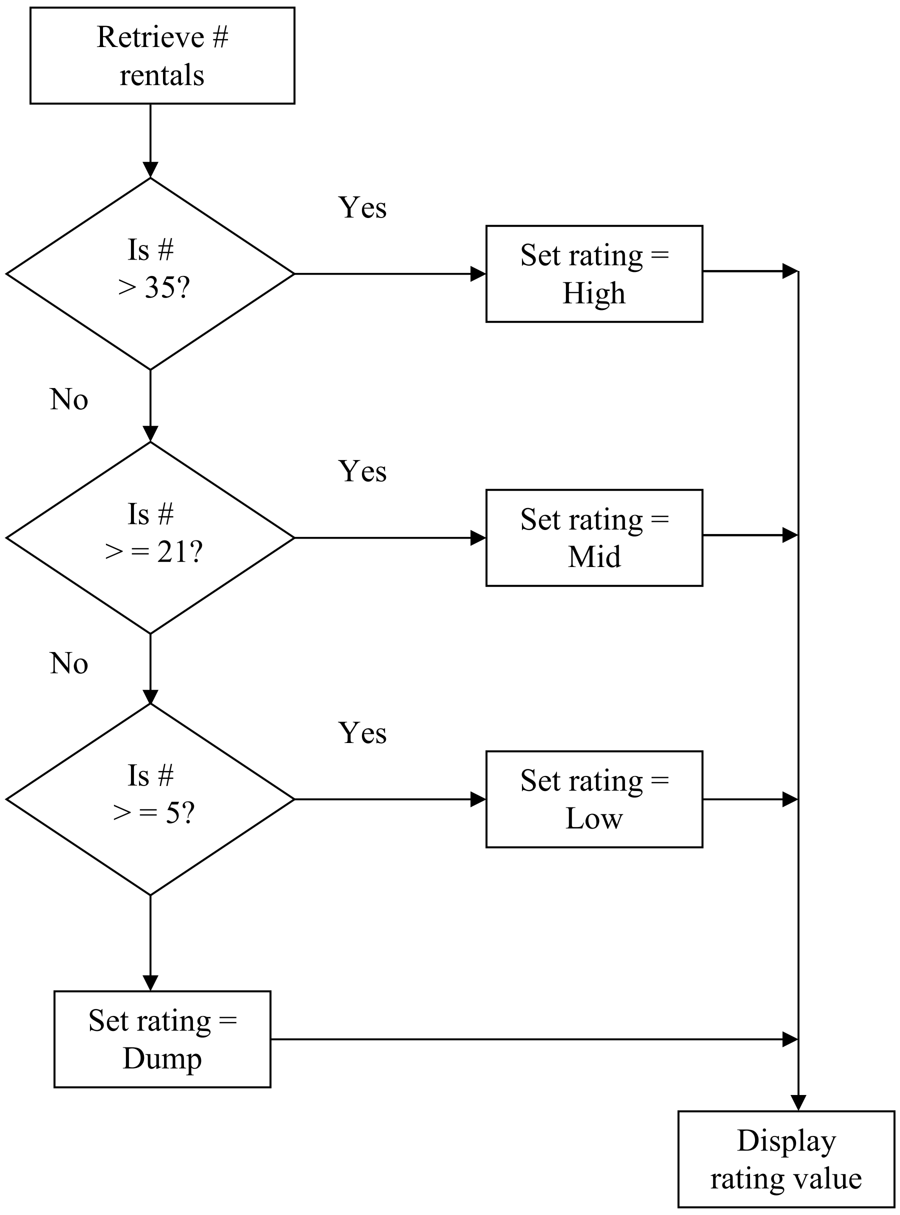
Case Projects

Case 2-1

Answers will vary. Examples of sites students might use include the following:

- *www.nos.org/htm/basic2.htm*
- *www.smartdraw.com/tutorials/flowcharts/tutorial_01.htm*
- *http://users.evtek.fi/~jaanah/IntroC/DBeech/3gl_flow.htm*
- *www.rff.com/how_to_draw_a_flowchart.htm*
-

Case 2-2



```
DECLARE
  lv_cnt_num NUMBER(3) := 18;
  lv_rating_txt VARCHAR2(4);
BEGIN
  IF lv_cnt_num > 35 THEN
    lv_rating_txt := 'High';
   ELSIF lv_cnt_num >= 21 THEN
    lv_rating_txt := 'Mid';
   ELSIF lv_cnt_num >= 5 THEN
    lv_rating_txt := 'Low';
   ELSE
    lv_rating_txt := 'Dump';
  END IF;
  DBMS_OUTPUT.PUT_LINE('Rating = '||lv_rating_txt);
END;
```

Chapter 2

Basic PL/SQL Block Structure

Content Listing

Chapter Overview

This chapter introduces basic PL/SQL block structure and logical processing. An initial discussion of programming logic and flowcharts is included for those that are new to programming. The first blocks created will introduce declaring and using scalar variables. Then students will be exposed to logical processing focusing on two types of structures: conditional processing with IF or CASE statements and using loops to repeat statements. The Brewbean's challenge of calculating tax costs based on the shipping state will assist students in understanding the need for logical processing statements to perform decision-making within an application.

Chapter Objectives

After completing this chapter, you should be able to understand:

- Programming fundamentals
- The PL/SQL block
- How to define and declare variables
- How to initialize and manage variable values
- The NOT NULL and CONSTANT variable options
- 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

Chapter by Section

➢ Programming Fundamentals

Instructor Notes:

All programs are written to accomplish a specific sequence of events. The logic and sequence of events needed should be identified prior to coding using some type of pseudocode or flowchart. The steps identified should include all data handling and decision making necessary.

Troubleshooting Tips	Identify a coding task and develop the flowchart in steps to address required processing.

Quick Quizzes	1. What type of structure do programmers use to repeat actions? Answer: Loops 2. What type of structure is used to determine what actions occur at run time? Answer: Decision structure 3. What method is used to graphically display the sequence of actions in a program? Answer: Flowcharting

➢ PL/SQL Block Structure

Instructor Notes:

A PL/SQL block consists of four main segments: DECLARE, BEGIN, EXCEPTION, and END. All variables, cursors, and types used throughout the block must be declared in the DECLARE section. The BEGIN section contains all the logical processing statements and SQL statements to interact with the database. The EXCEPTION section contains handlers that will control what the application will do if errors occur. In Chapters 2 and 3, anonymous blocks (which are not named or stored) will be created and executed in SQL*Plus to learn PL/SQL coding techniques.

Troubleshooting Tips	Run a simple PL/SQL block such as the one listed below to highlight that blocks executed in SQL*Plus must close with a forward slash on the last line to instruct the system to execute the block.
	`DECLARE`
	`  lv_day DATE;`
	`BEGIN`
	`   Lv_day := SYSDATE;`
	`END;`

Quick Quizzes	1. How will errors in a PL/SQL block be managed? Answer: Exception handlers 2. What section of the PL/SQL block contains variable declarations? Answer: DECLARE section 3. What section of the PL/SQL block can contain SQL statements? Answer: BEGIN section

➢ Working with Scalar Variables

Instructor Notes:

Scalar data types can only hold a single value. Value types of character, numeric, date, and Boolean can be handled by scalar variables. A variable declaration requires a variable name and data type. The keyword DEFAULT or the assignment symbol of := can be used in a variable declaration to initialize a variable to a value. If a variable is not initialized, then it contains a NULL value upon entrance into the BEGIN section of the block. The NOT NULL option requires the variable to always contain a value and, therefore, the variable must be initialized. The CONSTANT option protects a variable value from being modified. Using the DBMS_OUTPUT.PUT_LINE procedure provides an easy method to check variable values.

Troubleshooting Tips	Attempt executing a PL/SQL block that contains a DBMS_OUTPUT.PUT_LINE statement without first enabling DBMS_OUTPUT to demonstrate that no error is raised and nothing is displayed from the statement.
Quick Quizzes	1. How many values can a scalar variable contain? Answer: One 2. What variable declaration option prevents a variable value from being changed? Answer: CONSTANT 3. What is required in a variable declaration? Answer: A name and data type 4. What procedure enables a developer to check the values of variables during execution? Answer: DBMS_OUTPUT.PUT_LINE

➢ Initializing Variables

Instructor Notes:

It may be desirable for a variable to contain a value as processing the block begins. The keyword DEFAULT or the assignment symbol of := can be used in a variable declaration to initialize a variable to a value. If a variable is not initialized, then it contains a NULL value upon entrance into the BEGIN section of the block.

Troubleshooting Tips	Attempt executing a PL/SQL block that contains one variable initialized to a value and another that is not initialized. Execute the block using DBMS_OUTPUT.PUT_LINE statements to demonstrate the initial values in the variables.

Quick Quizzes	1. What is accomplished in variable initialization? Answer: A variable is assigned a value when it is declared 2. What is used in a variable declaration to accomplish initialization? Answer: Keyword of DEFAULT or the := symbol 3. Provide an example of why a variable may need to be initialized. Answer: (Answers will vary) To store the current date to for a data calculation

➢ Variable Options

Instructor Notes:

Declared variables can use two options to control requirements on the initialized values. The NOT NULL option forces a variable to contain a value and, therefore, it must be initialized. The CONSTANT option will prevent an initialized variable from being modified in the block.

Troubleshooting Tips	Attempt executing a PL/SQL block that contains one variable for each of the two options introduced. Execute the block using DBMS_OUTPUT.PUT_LINE statements to demonstrate the initial values in the variables. Also add a statement in the block that attempts to modify the value of the variable using the CONSTANT option to demonstrate an error will be raised.
Quick Quizzes	1. What variable option will not allow a variable's value to be altered? Answer: CONSTANT 2. What variable option requires that the variable always contain a value? Answer: NOT NULL 3. Does the variable CONSTANT option require the variable to be initialized? Answer: Yes

➢ Performing Calculations

Instructor Notes:

A variable can hold the result of a mathematical operation such as multiplication. An assignment statement is used to place a value in a variable. In this case, the value will be a calculation. Basic manipulations introduced in SQL can be used, such as addition (+), subtraction (-), multiplication, (*) and division (/).

Troubleshooting Tips	Demonstrate a block containing a calculation emphasizing the use of the := symbol for the assignment statement.
Quick Quizzes	1. What is an assignment statement? Answer: A statement that places a value into a variable 2. Can variables be used as part of a calculation? Answer: Yes 3. What symbol is used in an assignment statement? Answer: :=

➢ Using SQL Functions with Variables

Instructor Notes:

SQL contains many single row functions that can perform a variety of data manipulation tasks. PL/SQL variable values can be derived from a result of these functions.

Troubleshooting Tips	Execute a PL/SQL block that contains one variable. Use the MONTHS_BETWEEN function in an assignment statement for this variable. Execute the block using DBMS_OUTPUT.PUT_LINE statement to demonstrate the resulting value.
Quick Quizzes	1. What type of SQL functions can be used in a PL/SQL assignment statement to assign a value to a scalar variable? Answer: Single row functions 2. If an assignment statement uses the UPPER function, what data type is the variable? Answer: Character 3. What function would be required if a numeric value needs to be manipulated to contain a dollar sign, commas, and a decimal point? Answer: TO_CHAR

➢ Decision or Control Structures

Instructor Notes:

Conditional processing or the ability to check conditions and determine which statements should be processed is available in PL/SQL via two features: If and CASE statements. CASE statements were added to the PL/SQL language arsenal in Oracle9i based on user input, as many programmers are familiar with CASE statements from other programming languages. The choice between which construct to use is more a matter of preference. The If statements expand beyond simple condition checks to more complex checks by adding multiple conditions using the ELSIF clause. It is critical to recognize that If clauses are evaluated from top down, and once a true condition is discovered, the associated statements are processed and the If statement is ended. The CASE statement also processes top down until a true condition is found and, in addition, has two special formats named searched CASE and CASE expression.

Troubleshooting Tips	The misspelling of ELSIF is one of the most common errors of PL/SQL beginners. Everyone wants to add an 'E' and spell it as ELSEIF. Another common mistake is using a := operator in an If clause. Remind students that the If clause is not an assignment.
Quick Quizzes	1. Does an IF statement require at least one conditional clause to execute? Answer: No 2. What clause can be used in an If statement to ensure that some processing always occurs? Answer: ELSE 3. A case expression returns a value in a(n) _______________ statement. Answer: Assignment 4. What is used to check multiple conditions in an If clause? Answer: Logical operators (AND, OR) 5. Are the capabilities provided by IF and CASE statements are equivalent? Answer: Yes

➢ Looping Constructs

Instructor Notes:

Loop constructs allow developers to repeat a set of statements, which is particularly important for processing groups of records. The Brewbean's application requirement to calculate different tax rates for equipment and coffee items that are in an order highlights the need to repeat an operation for each record in a group of records, which represents a shopping cart. Three forms of loops exist: Basic, While, and For. The main difference of the forms is how the loop is ended.

Troubleshooting Tips	The most critical portion of a loop is the determination of when the loop processing ends. Providing an invalid condition to end a loop can lead to the most dreaded of errors: the infinite loop.
Quick Quizzes	1. What statement is used to end a Basic loop? Answer: Exit 2. When does a While loop end processing? Answer: When the condition in the While clause is false 3. When does a For loop end processing? Answer: When the counter reaches the last number in the range, the loop runs one more time and then ends

➢ Working with Nested Statements

Instructor Notes:

IF and Loop structures may be nested to branch decision making or looping steps. Using nested IF statements can make decision making steps easier to designate rather than combining logical operators in the IF statement. It is important to understand how the processing flows when nested statements are used.

Troubleshooting Tips	Identify the processing flow with nested IF and Loop statements.

Quick Quizzes	1. How many times will a statement execute if it is embedded in a nested loop which loops 5 times for each time the outer loop executes? Answer: Number of outer loop executions x 5 2. At what point will a nested IF statement return the processing to the outer IF statement? Answer: When a TRUE condition is reached or the ELSE clause executes

Classroom Activities

1. Chapter 1 instructed students to execute the initial Brewbean's database creation script that will prepare the database for this chapter. Be sure students have executed the script properly.

2. Be sure students execute the More Movies database script as outlined in Chapter 1 before completing the More Movies case study.

Discussion Questions

1. Are variables, decision structures, and loops used in other programming languages?

2. How are the PL/SQL data types similar or different from variable types used in other languages?

Projects to Assign

1. Describe a processing scenario/screen required by Brewbean's. Require students to create a flowchart outlining the processing steps needed.

2. Refer students to the PL/SQL reference provided on OTN to determine other data types that are available for PL/SQL variables.

Key Terms

anonymous blocks -- blocks of code that are not stored for reuse and, as far as the Oracle server is concerned, no longer exist after being executed

case expression -- evaluates conditions within an assignment statement

control structures -- perform conditional logic to determine which statements are executed at run time

looping constructs -- allows the repeated processing of a desired segment of code

scalar variables -- variables that can hold a single value. The common data types used for scalar variables include character, numeric, date, and Boolean.

searched case statement -- evaluates separate conditions fully identified in WHEN clauses