**Assignment 2-1 Using scalar variables**

1. **Create a PL/SQL. block that contains the following variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data Type** | **Option** | **Initial Value** |
| **lv\_test\_date** | **DATE** |  | **December 10,1007** |
| **lv\_test\_num** | **NUMBER(3)** | **CONSTANT** | **10** |
| **lv\_test\_txt** | **VARCHAR2(10)** |  |  |

**Assign your last name as the value of the text variable in the executable section of the block. Include statements in the block to display the value of each variable on the screen.**

|  |
| --- |
| DECLARE  lv\_test\_date DATE := TO\_DATE('1007-12-10', 'YYYY-MM-DD');  lv\_test\_num CONSTANT NUMBER(3) := 10;  lv\_test\_txt VARCHAR2(10);  BEGIN  -- Assign your last name to the text variable  lv\_test\_txt := 'Gurung'; -- Replace 'Smith' with your actual last name  -- Display the values of each variable  DBMS\_OUTPUT.PUT\_LINE('Test Date: ' || TO\_CHAR(lv\_test\_date, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('Test Number: ' || lv\_test\_num);  DBMS\_OUTPUT.PUT\_LINE('Test Text: ' || lv\_test\_txt);  END;  / |

1. The Brewbean’s application needs a block that determines if a customer is rated high, mid, or low based on his or her total purchases. The block must determine the rating and display the results on the screen. The code rates the customer HIGH if total purchases and then displays the $200, MID if greater than $100, and LOW if $100 or lower. Develop a flowchart to outline the conditional processing steps needed to handle this scenario.

|  |
| --- |
| DECLARE  total\_purchases NUMBER;  rating VARCHAR2(10);  BEGIN  -- Example input for total purchases (replace with actual input as needed)  total\_purchases := 150; -- Change this value to test different scenarios  -- Determine the rating based on total purchases  IF total\_purchases > 200 THEN  rating := 'HIGH';  ELSIF total\_purchases > 100 THEN  rating := 'MID';  ELSE  rating := 'LOW';  END IF;  -- Display the rating  DBMS\_OUTPUT.PUT\_LINE('Customer Rating: ' || rating);  END;  / |

1. Create a block using an IF statement to perform the action described in Assignment 2-2. Use a scalar variable for the total purchase and initialize the variable to different values to test your block.

|  |
| --- |
| DECLARE  total\_purchases NUMBER := 150; -- Initialize with different values to test (e.g., 50, 150, 250)  rating VARCHAR2(10);  BEGIN  -- Determine the rating based on total purchases  IF total\_purchases > 200 THEN  rating := 'HIGH';  ELSIF total\_purchases > 100 THEN  rating := 'MID';  ELSE  rating := 'LOW';  END IF;  -- Display the rating  DBMS\_OUTPUT.PUT\_LINE('Customer Rating: ' || rating);  END;  / |

1. Create a block using a CASE statement to perform the action described in Assignment 2-2. Use a scalar variable for the total purchase amount and initialize the variable to different values to test your block.

|  |
| --- |
| DECLARE  total\_purchases NUMBER := 150; -- Initialize with different values to test (e.g., 50, 150, 250)  rating VARCHAR2(10);  BEGIN  -- Determine the rating based on total purchases using a CASE statement  rating := CASE  WHEN total\_purchases > 200 THEN 'HIGH'  WHEN total\_purchases > 100 THEN 'MID'  ELSE 'LOW'  END;  -- Display the rating  DBMS\_OUTPUT.PUT\_LINE('Customer Rating: ' || rating);  END;  / |

1. Create an SQL\*Plus variable and modify the block in assignments 2-3 to use a host variable to provide the total purchase amount to the block.

|  |
| --- |
| VARIABLE total\_purchases NUMBER;  -- Set the value for total\_purchases variable  EXEC :total\_purchases := 150; -- Change this value to test different scenarios  DECLARE  rating VARCHAR2(10);  BEGIN  -- Determine the rating based on the total purchases using a CASE statement  rating := CASE  WHEN :total\_purchases > 200 THEN 'HIGH'  WHEN :total\_purchases > 100 THEN 'MID'  ELSE 'LOW'  END;  -- Display the rating  DBMS\_OUTPUT.PUT\_LINE('Customer Rating: ' || rating);  END;  / |

1. Create a block using a loop that will determine the number of items that can be purchased based on the price of the item and the total amount available to spend. Include one initialized variable to represent the price amount and another to represent the total available to spend. (We could solve using division, but we need to practice using loop structures.) The block needs to include statements to display the total number of items that can be purchased and the total amount that would be spent.

|  |
| --- |
| DECLARE  item\_price NUMBER := 15; -- Price of a single item  total\_available NUMBER := 100; -- Total amount available to spend  total\_spent NUMBER := 0; -- Total amount spent  items\_purchased NUMBER := 0; -- Number of items purchased  BEGIN  -- Loop until the total available amount is less than the item price  WHILE total\_available >= item\_price LOOP  total\_spent := total\_spent + item\_price; -- Increment total spent  items\_purchased := items\_purchased + 1; -- Increment item count  total\_available := total\_available - item\_price; -- Decrement available amount  END LOOP;  -- Display the results  DBMS\_OUTPUT.PUT\_LINE('Total Items Purchased: ' || items\_purchased);  DBMS\_OUTPUT.PUT\_LINE('Total Amount Spent: ' || total\_spent);  END;  / |

1. Brew beans' will determine shipping costs based on the number of items ordered and club membership status. The rates applicable are shown in the table below. Develop a flowchart to outline the conditional processing steps needed to handle this scenario.

|  |  |  |
| --- | --- | --- |
| Quantity of Items | Nonmember Shipping Cost | Member Shipping Cost |
| Up to 3 | $5.00 | $3.00 |
| 4-6 | $7.50 | $5.00 |
| 7-10 | $10.00 | $7.00 |
| Over 10 | $12.00 | $9.00 |