**PL/SQL Fundamental Exercises**

1. Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

SET SERVEROUTPUT ON;

DECLARE

v\_incentive NUMBER(8,2);

BEGIN

SELECT SALARY\*0.2

INTO v\_incentive

FROM TBLEMPLOYEES

WHERE EMPLOYEE\_ID = &EMPLOYEE\_ID;

DBMS\_OUTPUT.PUT\_LINE('Incentive of employee is '||v\_incentive);

END;

1. Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

SET SERVEROUTPUT ON;

DECLARE

v\_unquoted NUMBER(8,2):= 100;

"V\_QUOTED" NUMBER(8,2) := 200;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('UNQUOTED, MATCHING CASE - ' || v\_unquoted);

DBMS\_OUTPUT.PUT\_LINE('UNQUOTED, NOT MATCHING CASE - ' || V\_Unquoted);

DBMS\_OUTPUT.PUT\_LINE('QUOTED, MATCHING CASE' || "V\_QUOTED");

-- DBMS\_OUTPUT.PUT\_LINE('QUOTED, NOT MATCHING CASE' || "v\_quoted"); --- WILL THROW AN ERROR

END;

1. Write a PL/SQL block to show a reserved word can be used as a user-define identifier.

SET SERVEROUTPUT ON;

DECLARE

"IF" NUMBER(8,2):= 100;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('VALUE OF "IF" - ' || "IF");

END;

1. Write a PL/SQL block to show the result to neglect double quotation marks in reserved word identifier.

SET SERVEROUTPUT ON;

DECLARE

"IF" NUMBER(8,2):= 100;

BEGIN

-- DBMS\_OUTPUT.PUT\_LINE('VALUE OF IF - ' || IF); --will throw an error

END;

1. Write a PL/SQL block to show the result to neglect the case sensitivity of a user defined identifier which is also a reserved word.

SET SERVEROUTPUT ON;

DECLARE

"IF" NUMBER(8,2):= 100;

BEGIN

--- DBMS\_OUTPUT.PUT\_LINE('VALUE OF IF - ' || "iF"); --throws error

END;

1. Write a PL/SQL block to explain single and multiline comments.

SET SERVEROUTPUT ON;

DECLARE

"IF" NUMBER(8,2):= 100;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Demonstration of comments (will not be displayed in the output)');

-- This is a single line comment.

/\*

This is a

multi line

comment

.

\*/

END;

1. Write PL/SQL blocks to show the declaration of variables.

SET SERVEROUTPUT ON;

DECLARE

v\_employee\_id NUMBER(8,2) :=&employee\_id;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Employee ID is '||v\_employee\_id);

END;

1. Write PL/SQL blocks to show the scope and visibility of local and global identifiers.

SET SERVEROUTPUT ON;

DECLARE

v\_num NUMBER(8,2) := 100; --Global variable

BEGIN

DBMS\_OUTPUT.PUT\_LINE('v\_num '||v\_num); --value displayed : 100

DECLARE

v\_num NUMBER(8,2) := 200;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('v\_num (inside the nested block)'||v\_num); --value displayed : 200

END;

DBMS\_OUTPUT.PUT\_LINE('v\_num (after the block)'||v\_num); --value displayed : 100

END;

1. Write a PL/SQL block to show a valid case-insensitive reference to a quoted and without quoted user-defined identifier.

SET SERVEROUTPUT ON;

DECLARE

"V\_QUOTED" NUMBER(8,2) := 200;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('UNQUOTED REFERENCE , MATCHING CASE - ' || V\_QUOTED);

DBMS\_OUTPUT.PUT\_LINE('UNQUOTED REFERENCE , NOT MATCHING CASE - ' || v\_quoted);

DBMS\_OUTPUT.PUT\_LINE('QUOTED REFERENCE , MATCHING CASE - ' || "V\_QUOTED");

-- DBMS\_OUTPUT.PUT\_LINE('QUOTED REFERENCE , NOT MATCHING CASE - ' || "v\_quoted"); --- WILL THROW AN ERROR

END;

1. Write a PL/SQL block to adjust the salary of the employee whose ID 122. (Read id from the user)

SET SERVEROUTPUT ON;

DECLARE

v\_salary NUMBER(8,2);

BEGIN

SELECT SALARY

INTO v\_salary

FROM TBLEMPLOYEES

WHERE EMPLOYEE\_ID = &EMPLOYEE\_ID;

DBMS\_OUTPUT.PUT\_LINE('Original salary of employee is '||v\_salary);

v\_salary := v\_salary\*2/3-1+10;

DBMS\_OUTPUT.PUT\_LINE('Modified salary of employee is '||v\_salary);

END;

1. Write a PL/SQL block to show the operator precedence and parentheses in several more complex expressions.

SET SERVEROUTPUT ON;

DECLARE

v\_num NUMBER(8,2) :=10;

v\_num1 NUMBER(8,2);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Original num is '||v\_num);

v\_num1 := v\_num\*2/4-1+10;

DBMS\_OUTPUT.PUT\_LINE('v\_num\*2/4-1+10 is '||v\_num1);

v\_num1 := v\_num\*2/(4-1)+10;

DBMS\_OUTPUT.PUT\_LINE('v\_num\*2/(4-1)+10 is '||v\_num1);

END;

1. Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.
2. Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show OR operator returns TRUE if either operand is TRUE.
3. Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show NOT operator returns the opposite of its operand, unless the operand is NULL.
4. Write a PL/SQL block to describe the usage of NULL values in equal comparison, unequal comparison and NOT NULL equals NULL comparison.
5. Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

**Practice 1**

The labs folder will be your working directory. You can save your scripts in the labs folder. Please take the instructor’s help to locate the labs folder for this course. The solutions for all practices are in the soln folder.

1. Which of the following PL/SQL blocks execute successfully?

a. BEGIN  
END;

b. DECLARE  
amount INTEGER(10);  
END;

c. DECLARE  
BEGIN  
END;

**d. DECLARE  
amount INTEGER(10);  
BEGIN  
DBMS\_OUTPUT.PUT\_LINE(amount);  
END;**

2. Create and execute a simple anonymous block that outputs “Hello World.” Execute and save this script as lab\_01\_02\_soln.sql.

SET SERVEROUTPUT ON;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

**Practice 2**

1. Identify valid and invalid identifier names:

a. today valid

b. last\_name valid

c. today’s\_date invalid

d. Number\_of\_days\_in\_February\_this\_year valid

e. Isleap$year valid

f. #number invalid

g. NUMBER# valid

h. number1to7 valid

2. Identify valid and invalid variable declaration and initialization:

a. number\_of\_copies PLS\_INTEGER; valid

b. printer\_name constant VARCHAR2(10); valid

c. deliver\_to VARCHAR2(10):=Johnson; invalid

d. by\_when DATE:= CURRENT\_DATE+1; valid

3. Examine the following anonymous block and choose the appropriate statement.

DECLARE

v\_fname VARCHAR2(20);

v\_lname VARCHAR2(15) DEFAULT 'fernandez';

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_fname ||' ' ||v\_lname);

END;

/

**a. The block executes successfully and print “fernandez.”**

b. The block returns an error because the fname variable is used without initializing.

c. The block executes successfully and print “null fernandez.”

d. The block returns an error because you cannot use the DEFAULT keyword to initialize a variable of type VARCHAR2.

e. The block returns an error because the v\_fname variable is not declared.

* + 1. Create an anonymous block. In SQL Developer, load the lab\_01\_02\_soln.sql script, which you created in question 2 of practice 1.

a. Add a declarative section to this PL/SQL block. In the declarative section, declare the following variables:

1. Variable v\_today of type DATE. Initialize today with SYSDATE.

2. Variable v\_tomorrow of type today. Use %TYPE attribute to declare this variable.

b. In the executable section, initialize the tomorrow variable with an expression, which calculates tomorrow’s date (add one to the value in today). Print the value of today and tomorrow after printing “Hello World.”

c. Execute and save this script as lab\_02\_04\_soln.sql. Sample output is as follows:

SET SERVEROUTPUT ON;

DECLARE

v\_today DATE := SYSDATE;

v\_tomorrow v\_today%TYPE;

BEGIN

v\_tomorrow := v\_today +1;

DBMS\_OUTPUT.PUT\_LINE('Hello World');

DBMS\_OUTPUT.PUT\_LINE('Today: '||v\_today);

DBMS\_OUTPUT.PUT\_LINE('Tomorrow: '||v\_tomorrow);

END;

* + 1. Edit the lab\_02\_04\_soln.sql script.

a. Add code to create two bind variables.   
Create bind variables b\_basic\_percent and b\_pf\_percent of type NUMBER.

b. In the executable section of the PL/SQL block, assign the values 45 and 12 to b\_basic\_percent and b\_pf\_percent, respectively.

c. Terminate the PL/SQL block with “/” and display the value of the bind variables by using the PRINT command.

* + - 1. Execute and save your script file as lab\_02\_05\_soln.sql. Sample output is as follows:

SET SERVEROUTPUT ON;

SET AUTOPRINT ON;

VARIABLE b\_basic\_percent NUMBER;

VARIABLE b\_pf\_percent NUMBER;

DECLARE

v\_today DATE := SYSDATE;

v\_tomorrow v\_today%TYPE;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

v\_tomorrow := v\_today +1;

DBMS\_OUTPUT.PUT\_LINE('Today: '||v\_today);

DBMS\_OUTPUT.PUT\_LINE('Tomorrow: '||v\_tomorrow);

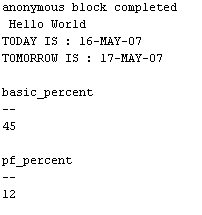
:b\_basic\_percent := 45;

:b\_pf\_percent := 12;

--DBMS\_OUTPUT.PUT\_LINE(:b\_basic\_percent);

-- DBMS\_OUTPUT.PUT\_LINE(:b\_pf\_percent);

END;



**Practice 3**

**PL/SQL Block**

DECLARE  
 v\_weight NUMBER(3) := 600;  
 v\_message VARCHAR2(255) := 'Product 10012';  
BEGIN  
 DECLARE  
 v\_weight NUMBER(3) := 1;  
 v\_message VARCHAR2(255) := 'Product 11001';  
 v\_new\_locn VARCHAR2(50) := 'Europe';  
 BEGIN  
 v\_weight := v\_weight + 1;  
 v\_new\_locn := 'Western ' || v\_new\_locn;  
  
 END;  
 v\_weight := v\_weight + 1;  
 v\_message := v\_message || ' is in stock';  
 v\_new\_locn := 'Western ' || v\_new\_locn;  
  
END;  
/

1. Evaluate the preceding PL/SQL block and determine the data type and value of each of the following variables according to the rules of scoping.

a. The value of v\_weight at position 1 is: 1

b. The value of v\_new\_locn at position 1 is: ‘Europe’

c. The value of v\_weight at position 2 is: 2

d. The value of v\_message at position 2 is: ‘Product 10012 is in stock’

e. The value of v\_new\_locn at position 2 is: ‘Western Europe’

**Scope Example**

DECLARE  
 v\_customer VARCHAR2(50) := 'Womansport';  
 v\_credit\_rating VARCHAR2(50) := 'EXCELLENT';  
BEGIN  
 DECLARE  
 v\_customer NUMBER(7) := 201;  
 v\_name VARCHAR2(25) := 'Unisports';   
 BEGIN  
 v\_credit\_rating :='GOOD';   
 …  
 END;  
 …  
END;  
/

2. In the preceding PL/SQL block, determine the values and data types for each of the following cases.

a. The value of v\_customer in the nested block is: 201

b. The value of name in the nested block is: 'Unisports'

c. The value of v\_credit\_rating in the nested block is: ‘GOOD’

d. The value of v\_customer in the main block is: 'Womansport'

e. The value of name in the main block is: error

f. The value of v\_credit\_rating in the main block is: 'EXCELLENT'

3. Edit lab\_02\_05\_soln.sql.

a. Use single-line comment syntax to comment the lines that create the bind variables.

b. Use multiple-line comments in the executable section to comment the lines that assign values to the bind variables.

c. Declare the v\_basic\_percent and v\_pf\_percent variables and initialize them to 45 and 12, respectively. Also, declare two variables: v\_fname of type VARCHAR2 and size 15, and v\_emp\_sal of type NUMBER and size 10.

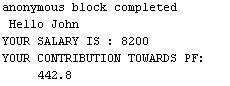
d. Include the following SQL statement in the executable section:

SELECT first\_name, salary   
INTO v\_fname, v\_emp\_sal FROM employees   
WHERE employee\_id=110;

e. Change the line that prints “Hello World” to print “Hello” and the first name. You can comment the lines that display the dates and print the bind variables, if you want to.

f. Calculate the contribution of the employee toward provident fund (PF).  
PF is 12% of the basic salary and basic salary is 45% of the salary. Use the bind variables for the calculation. Try and use only one expression to calculate the PF. Print the employee’s salary and his contribution toward PF.

g. Execute and save your script as lab\_03\_03\_soln.sql. Sample output is as follows:



SET SERVEROUTPUT ON;

SET AUTOPRINT OFF;

--VARIABLE b\_basic\_percent NUMBER;

--VARIABLE b\_pf\_percent NUMBER;

DECLARE

v\_today DATE := SYSDATE;

v\_tomorrow v\_today%TYPE;

v\_basic\_percent NUMBER := 45;

v\_pf\_percent NUMBER :=12;

v\_fname VARCHAR2(15);

v\_emp\_sal NUMBER(8,2);

v\_cont\_pf NUMBER(8,2);

BEGIN

SELECT first\_name, salary

INTO v\_fname, v\_emp\_sal FROM TBLEMPLOYEES

WHERE employee\_id=110;

DBMS\_OUTPUT.PUT\_LINE('Hello '||v\_fname);

v\_cont\_pf := v\_pf\_percent\*(v\_emp\_sal\*v\_basic\_percent/100)/100;

DBMS\_OUTPUT.PUT\_LINE('YOUR SALARY IS '||v\_emp\_sal);

DBMS\_OUTPUT.PUT\_LINE('YOUR CONTRIBUTION TOWARDS PF: '||v\_cont\_pf);

/\*

v\_tomorrow := v\_today +1;

DBMS\_OUTPUT.PUT\_LINE('Today: '||v\_today);

DBMS\_OUTPUT.PUT\_LINE('Tomorrow: '||v\_tomorrow);

:b\_basic\_percent := 45;

:b\_pf\_percent := 12;

DBMS\_OUTPUT.PUT\_LINE(:b\_basic\_percent);

DBMS\_OUTPUT.PUT\_LINE(:b\_pf\_percent);

\*/

END;

**Practice 4**

1. Create a PL/SQL block that selects the maximum department ID in the departments table and stores it in the v\_max\_deptno variable. Display the maximum department ID.

a. Declare a variable, v\_max\_deptno, of type NUMBER in the declarative section.

b. Start the executable section with the BEGIN keyword and include a SELECT statement to retrieve the maximum department\_id from the departments table.

c. Display v\_max\_deptno and end the executable block.

d. Execute and save your script as lab\_04\_01\_soln.sql. Sample output is as follows:

DECLARE

v\_max\_deptno NUMBER;

BEGIN

SELECT MAX(DEPARTMENT\_ID)

INTO v\_max\_deptno

FROM TBLDEPARTMENTS;

DBMS\_OUTPUT.PUT\_LINE('Maximum department id is ' ||v\_max\_deptno);

END;

1. Modify the PL/SQL block you created in exercise 1 to insert a new department in the departments table.

a. Load the lab\_04\_01\_soln.sql script. Declare two variables:   
 v\_dept\_name of type departments.department\_name  
 v\_dept\_id of type NUMBER  
Assign “Education” to v\_dept\_name in the declarative section.

b. You have already retrieved the current maximum department ID from the departments table. Add 10 to it and assign the result to v\_dept\_id.

c. Include an INSERT statement to insert data into the department\_name, department\_id, and location\_id columns of the departments table.   
Use values in v\_dept\_name and v\_dept\_id for department\_name and department\_id, respectively, and use NULL for location\_id.

d. Use the SQL attribute SQL%ROWCOUNT to display the number of rows that are affected.

e. Execute a SELECT statement to check whether the new department is inserted. You can terminate the PL/SQL block with “/” and include the SELECT statement in your script.

f. Execute and save your script as lab\_04\_02\_soln.sql. Sample output is as follows:

SET SERVEROUTPUT ON;

DECLARE

v\_max\_deptno NUMBER;

v\_dept\_name TBLDEPARTMENTS.DEPARTMENT\_NAME%TYPE := 'Education';

v\_dept\_id NUMBER;

v\_did NUMBER;

v\_dname v\_dept\_name%TYPE;

BEGIN

SELECT MAX(DEPARTMENT\_ID)

INTO v\_max\_deptno

FROM TBLDEPARTMENTS;

DBMS\_OUTPUT.PUT\_LINE('Maximum department id is ' ||v\_max\_deptno);

v\_dept\_id := v\_max\_deptno + 10;

INSERT INTO TBLDEPARTMENTS(DEPARTMENT\_NAME, DEPARTMENT\_ID, LOCATION\_ID) VALUES (v\_dept\_name, v\_dept\_id, NULL);

DBMS\_OUTPUT.PUT\_LINE('SQL%ROWCOUNT GIVES '|| SQL%ROWCOUNT);

SELECT DEPARTMENT\_ID , DEPARTMENT\_NAME INTO v\_did, v\_dname FROM TBLDEPARTMENTS WHERE DEPARTMENT\_ID = v\_dept\_id;

DBMS\_OUTPUT.PUT\_LINE(v\_did||' '||v\_dname);

END;

/





3. In exercise 2, you set location\_id to NULL. Create a PL/SQL block that updates the location\_id to 3000 for the new department. Use the bind variable dept\_id to update the row.  
**Note:** Skip step (a) if you have not started a new session for this practice.

a. If you have started a new session, delete the department that you have added to the departments table and execute the lab\_04\_02\_soln.sql script.

b. Start the executable block with the BEGIN keyword. Include the UPDATE statement to set the location\_id to 3000 for the new department (dept\_id = 280).

c. End the executable block with the END keyword. Terminate the PL/SQL block with “/” and include a SELECT statement to display the department that you updated.

d. Finally, include a DELETE statement to delete the department that you added.

* + - 1. Execute and save your script as lab\_04\_03\_soln.sql. Sample output is as follows:

anonymous block completed

1. DEPARTMENT\_ID DEPARTMENT\_NAME MANAGER\_ID LOCATION\_ID

------------- --------------- ----------- --------------

280 Education 3000

1 rows selected

1 rows deleted

SET SERVEROUTPUT ON;

VARIABLE b\_dept\_id NUMBER;

DECLARE

v\_max\_deptno NUMBER;

v\_dept\_name tbldepartments.department\_name%TYPE := 'Education';

v\_did NUMBER;

v\_dname v\_dept\_name%TYPE;

v\_loc tbldepartments.location\_id%TYPE;

BEGIN

SELECT

MAX(department\_id)

INTO v\_max\_deptno

FROM

tbldepartments;

dbms\_output.put\_line('Maximum department id is ' || v\_max\_deptno);

:b\_dept\_id := v\_max\_deptno + 10;

INSERT INTO tbldepartments (

department\_name,

department\_id,

location\_id

) VALUES (

v\_dept\_name,

:b\_dept\_id,

NULL

);

dbms\_output.put\_line('SQL%ROWCOUNT GIVES ' || SQL%rowcount);

SELECT

department\_id,

department\_name,

location\_id

INTO

v\_did,

v\_dname,

v\_loc

FROM

tbldepartments

WHERE

department\_id = :b\_dept\_id;

dbms\_output.put\_line(v\_did

|| ' '

|| v\_dname

|| ' '

|| v\_loc);

UPDATE tbldepartments

SET

location\_id = 3000

WHERE

department\_id = :b\_dept\_id;

SELECT

department\_id,

department\_name,

location\_id

INTO

v\_did,

v\_dname,

v\_loc

FROM

tbldepartments

WHERE

department\_id = :b\_dept\_id;

dbms\_output.put\_line(v\_did

|| ' '

|| v\_dname

|| ' '

|| v\_loc);

END;

rollback;