# Lab 7. PL/SQL Packages

**Point Distribution**:

| Question Number | Points |
|---|---|
| 1a, 1b, 1c, 1d, 1f, 2a, 2b, 2c, 2d, 2e, 2f, 2i | 0.75pt x 12 |
| 1e, 2g, 2h, 3c, 3f | 1pt x 5 |
| 3a, 3b, 3d, 3e | 1.5pt x 4 |
| Total | 20 |

**Submission**: A typed document with extension ".txt" OR ".sql" must be uploaded to the <<lab7>> submission link on Gradescope.

**Objectives**:

- Practice implementing PL/SQL Packages
- Practice implementing overloaded subroutines within PL/SQL packages
- There are no check offs for this lab. Lab attendance is optional as well.

## Questions

**Specification-only Packages**

Some Packages do not contain a body. Instead, they possess only a specification section.

These packages are typically used to hold values for constants.

**1a** Using the code provided below, create the MATH_CONSTANTS specification-only package.

Worksheet | Query Builder

```
1  CREATE OR REPLACE PACKAGE MATH_CONSTANTS
2  is
3     c_Pi CONSTANT NUMBER := acos(-1);
4  END MATH_CONSTANTS;
```

**1b** What is the acos() function? What is the value of acos(-1)? What is this value typically referred to?
NOTE: acos stands for arc-cosine

**1c** What is the code necessary to retrieve the value of c_Pi and print it to the screen as output? Include both the code and the results in your submission.

**1d** Try retrieving the value of c_Pi as you would call a function (using a SELECT statement). Can you do so?
Include both the code and the results in your submission.

**1e** Create a PL/SQL function named CALCULATE_AREA that accepts a single numeric parameter: p_radius. This module should:

- calculate the area of a circle of the specified radius (if you don't remember the formula, please look it up)

| | |
|---|---|
| | • use the value of c_Pi stored in the MATH_CONSTANTS package when calculating the area of the circle<br>• return a numeric value representing the area of the circle rounded to 3 digits |
| **1f** | Test CALCULATE_AREA with the following values:<br>• radius = 5<br>• radius = 10<br>• radius = 15<br><br>Include both the code and the results in your submission. |

**General PL/SQL Packages**

Most PL/SQL packages contain both a specification section and a body section. However, you may wish to test your functions and procedures prior to inserting them into the package.

**2a** Create the function PARSE_FIRSTNAME shown below.

Worksheet | Query Builder

```
1  CREATE OR REPLACE FUNCTION PARSE_FIRSTNAME
2    (p_fullname   supplier.contactname%type)
3  RETURN VARCHAR2
4  IS
5      v_firstname supplier.contactname%type;
6  BEGIN
7      v_firstname := SUBSTR(p_fullname, 1, instr (p_fullname, ' ', 1, 1) -1);
8    RETURN v_firstname;
9  END PARSE_FIRSTNAME;
10
```

**2b** What is the code necessary to use the PARSE_FIRSTNAME function to retrieve the first name of each contact in the Eagle Electronics SUPPLIER table?
Include both the code and the results in your submission.

**2c** Once we are confident that the function is working correctly, we can remove (drop) the "naked" function and subsequently incorporate it into a package.
Drop the PARSE_FIRSTNAME function.

**2d** As shown below, create the package specification for a package named PKG_FORMAT_EAGLE_DATA. This package (at least initially) contains a single function named PARSE_FIRSTNAME.

Worksheet | Query Builder

```
1  CREATE OR REPLACE PACKAGE PKG_FORMAT_EAGLE_DATA
2  IS
3
4    FUNCTION PARSE_FIRSTNAME
5      (p_fullname supplier.contactname%type)
6    RETURN VARCHAR2;
7
8  END PKG_FORMAT_EAGLE_DATA;
```

Notice the interface for the PARSE_FIRSTNAME function is the module type (function), module name (PARSE_FIRSTNAME), parameter list, and return type.

**2e** Subsequently create the package body for the PKG_FORMAT_EAGLE_DATA package as shown below.

Worksheet | Query Builder

```
1  CREATE OR REPLACE PACKAGE BODY PKG_FORMAT_EAGLE_DATA
2  IS
3      FUNCTION PARSE_FIRSTNAME
4        (p_fullname   supplier.contactname%type)
5      RETURN VARCHAR2
6      IS
7          v_firstname supplier.contactname%type;
8      BEGIN
9          v_firstname := SUBSTR(p_fullname, 1, instr (p_fullname, ' ', 1, 1) -1);
10       RETURN v_firstname;
11     END PARSE_FIRSTNAME;
12
13 END PKG_FORMAT_EAGLE_DATA;
```

Notice that the package body contains the full implementation details for the PARSE_FIRSTNAME function (unlike the package spec which contains no implementation details).

**2f** Test the PARSE_FIRSTNAME functionality of the PKG_FORMAT_EAGLE_DATA package using the contact name of the suppliers to Eagle Electronics.
Include both the code and the results in your submission.

**2g** Extend the package specification of the PKG_FORMAT_EAGLE_DATA package. Specifically, incorporate a 2nd public function named PARSE_LASTNAME into the package.

- The function should accept a character input in 'FirstName LastName' format
- The input should have the same data type with contact name from supplier table
- The function should return ONLY the last name of each contact

**2h** Extend the package body of the PKG_FORMAT_EAGLE_DATA package with the implementation details for the PARSE_LASTNAME function.

**2i** Using the contact name for each supplier to Eagle Electronics, test your PKG_FORMAT_EAGLE_DATA package.
Make certain to test **both** the PARSE_LASTNAME functionality (to confirm it works) as well as your PARSE_FIRSTNAME functionality (to confirm you have not broken anything).
Include both the code and the results in your submission.

**Implement Your Own PL/SQL Package**

**3** Create a package named PKG_SHAPE_CALCULATION. Include the following modules in this package, each of which is described in the sub-questions.

- A public function named RECTANGLE_AREA (see part a)
- A public function named RECTANGLE_AREA (see part b)
- A private constant named c_Pi (see part c)
- A public function named CIRCLE_AREA (see part d)
- A private function CHECK_VALUE that takes a numeric value and returns a boolean value that indicates whether the input is acceptable (see part e)

| | |
|---|---|
| **3a** | A public function named RECTANGLE_AREA that:<br>• accepts 2 numeric parameters: length and width<br>• verifies that the length and width are physically possible – use private function CHECK_VALUE<br>• calculates the area of a rectangle of the specified dimensions (if you don't remember the formula, please look it up)<br>• returns a numeric value representing this area<br>• incorporates a general exception handler to gracefully deal with exceptions |
| **3b** | A public function named RECTANGLE_AREA that:<br>• accepts 1 numeric parameter: length<br>• verifies that the length is physically possible – use private function CHECK_VALUE<br>• calculates the area of a square (e.g., a rectangle having sides of equal length) of the specified dimension (*NOTE: the function is named the same as in 5a, keep it this way*)<br>• returns a numeric value representing this area<br>• incorporates a general exception handler to gracefully deal with exceptions |
| **3c** | A private constant named c_Pi that uses the arccosine to determine the value of $\pi$. What is the benefit of making it private? |
| **3d** | A public function named CIRCLE_AREA that:<br>• accepts 1 numeric parameter: radius<br>• verifies that the radius is physically possible -– use private function CHECK_VALUE<br>• calculates the area of a circle having the specified radius<br>• returns a numeric value representing this area<br>• incorporates a general exception handler to gracefully deal with exceptions |
| **3e** | A private function named CHECK_VALUE that:<br>• accepts 1 numeric parameter which can used as a length, a width, or a radius<br>• returns a boolean value (True or False)<br>  ○ True should be returned if the value is above 0<br>  ○ False should be returned if the value is less than or equal to 0<br>• incorporates a general exception handler to gracefully deal with exceptions |
| **3f** | Test each of the modules in the PKG_SHAPE_CALCULATION package at least 3 times.<br>Make certain to include the following cases in the test for each module<br>• Positive number(s)<br>• Negative number(s)<br>• Zero(s)<br><br>Include both the code and the results in your submission. |