|  |
| --- |
| **Part A**  Create, compile, and execute a PL/SQL procedure from the Oracle SQL \* Plus  Create and PL/SQL functions. |
| **Aim:**   * Create, compile, and execute a PL/SQL procedure. * To understand %ROWTYPE Attribute * Learn and implement PL/SQL functions. |
| **Prerequisite:** Oracle, SQL |
| **Outcome:** Students will be able to write PL/SQL procedures |
| **Theory:**   * A PL/SQL procedure is a named [block](https://www.oracletutorial.com/plsql-tutorial/plsql-anonymous-block/) stored   + It is a reusable unit * The basic syntax of creating a procedure in PL/SQL is as follows: * Note that OR REPLACE option allows you to overwrite the current procedure with the new code. * PL/SQL procedures have to parts   + PL/SQL procedure header     - It  specifies procedure name and an optional parameter list.   + PL/SQL procedure body     - **Declarative part**     - **Executable part**     - **Exception-handling part** * A procedure begins with a header that specifies its name and an optional parameter list. * Parameters can in one of the following modes:   + In mode   + Out mode   + Inout mode * An IN parameter is read-only.   + You can reference an IN parameter inside a procedure, but you cannot change its value.   + Oracle uses IN as the default mode.     - It means that if you don’t specify the mode for a parameter explicitly, Oracle will use the IN mode. * An OUT parameter is writable.   + Typically, you set a returned value for the OUT parameter and return it to the calling program.   + Note that a procedure ignores the value that you supply for an OUT parameter. * An INOUT parameter is both readable and writable.   + The procedure can read and modify it.   **%ROWTYPE Attribute**   * The %ROWTYPE attribute provides a record type that represents a row in a database table. * The record can store an entire row of data selected from the table or fetched from a cursor or cursor variable. * Fields in a record and corresponding columns in a row have the same names and datatypes. * You can use the %ROWTYPE attribute in variable declarations as a datatype specifier. * Variables declared using %ROWTYPE are treated like those declared using a datatype name.   **Creating a PL/SQL procedure example**   * **The following procedure accepts a customer id and prints out the customer’s contact information including first name, last name, and email:**      * **Use / (forward slash) to compile procedures** * **Using the EXECUTE/EXEC keyword with procedure name to execute procedure**   + **EXECUTE procedure\_name( arguments);**   + **EXEC procedure\_name( arguments);** * **EXEC print\_contact(100);**     **PL/SQL Functions**  CREATE [OR REPLACE] FUNCTION function\_name  [(parameter\_name [IN | OUT | IN OUT] type [, ...])]  RETURN return\_datatype  {IS | AS}  BEGIN  < function\_body >  END [function\_name];  Where,   * *function-name* specifies the name of the function. * [OR REPLACE] option allows the modification of an existing function. * The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure. * The function must contain a **return** statement. * The *RETURN* clause specifies the data type you are going to return from the function. * *function-body* contains the executable part. * The AS keyword is used instead of the IS keyword for creating a standalone function. |
| **Practice Exercise**   |  |  | | --- | --- | | 1 | 1. Create a table with following schema:   Coustmer\_contact(cid number(10),cfname varchar(10),clname varchar(10),cemail varchar(10))   1. Create a PL/SQL procedure accepts a customer id and prints out the customer’s contact information including first name, last name, and email: | | 2 | Create a employee table with following schema  employee (e\_id number(10), e\_name varchar(10),e\_sales number(8), target number(8), salary number(8,2))  Create a PL/SQL procedure which follows following:   1. For a given e\_id it checks whether e\_sales greater than target or not.   If greater then  UPDATE employees SET salary = salary + bonus WHERE employee\_id = emp\_id;  Otherwise no update to salary   1. Bonus is target-sales. 2. An exception is raised if no records were found for given e\_id   Use inbuilt exception NO\_DATA\_FOUND | | 3 | Create a student table with following schema  student (s\_id number(10), s\_name varchar(10),s\_marks number(8))  Create a PL/SQL Program which follows following:   1. FOR a given s\_id select the s\_marks. 2. If marks greater than 90 print excellent 3. If marks greater than 80 but less than 90 print very good 4. If marks greater than 70 but less than 80 print good 5. If marks greater than 60 but less than 70 print fair 6. If marks greater than 50 but less than 60 print poor 7. For all other marks print fail   An exception is raised if no records were found | | 4 | Create a procedure which gets the name of the employee when the employee id is passed. Use out parameter to store and employee name and print it. | | 5 | Create a PL/SQL functions to calculate the area and perimeter (perimeter) of circle. | | 6 | Create a procedure and function to get the total number of employees in the employee table. | |
| **Instructions:**   1. Write and execute the query in Oracle SQL server/ SQL\* Plus. 2. Paste the snapshot of the output in input & output section. |
| **Part B** |
| **Code and Output:**  Perform the operation and paste the running code here. |
| **1.A** |
| **1.B** |
| **2.** |
| **3.A** |
| **3.B** |
| **4.** |
| **5.** |
| **6.** |
| **Observation & Learning:**  From this experiment, we observed and learned how to create, compile, and execute a PL/SQL procedure, understood %ROWTYPE Attribute in pl/sql and how to implement PL/SQL functions.in the **ORACLE DATABASE.** |
| **Conclusion:**  In this experiment, procedures and functions are created and implemented in pl/sql in the ORACLE DATABASE. |