PL/SQL function

- ***PL/SQL function*** is a named block that returns a value.
- PL/SQL functions are also known as subroutines or subprograms.
- To create a PL/SQL function, you use the following syntax

```
CREATE [OR REPLACE] FUNCTION {function_name}
[(
    {parameter_1} [IN] [OUT] {parameter_data_type_1},
    {parameter_2} [IN] [OUT] {parameter_data_type_2},...
    {parameter_N} [IN] [OUT] {parameter_data_type_N} )]

RETURN {return_datatype} IS
    --the declaration statements
BEGIN
    -- the executable statements
RETURN {return_data_type};
EXCEPTION
    -- the exception-handling statements
END;
```

PL/SQL function

- The {function_name} is the name of the function.
- Function name should start with a verb for example function convert_to_number.
- {parameter_name} is the name of parameter being passed to function along with parameter's data type {parameter_data_type}.
- There are three modes for parameters: IN,OUT and IN OUT.

PL/SQL function

- The IN mode is the default mode.
- You use the IN mode when you want the formal parameter is read-only.
- It means you cannot alter its value in the function.
- The IN parameter behaves like a constant inside the function.
- You can assign default value to the IN parameter or make it optional.

PL/SQL function

- The OUT parameters return values to the caller of a subprogram.
- An OUT parameter cannot be assigned a default value therefore you cannot make it optional.
- You need to assign values to the OUT parameter before exiting the function or its value will be NULL.
- From the caller subprogram, you must pass a variable to the OUT parameter.

PL/SQL function

- In the IN OUT mode, the actual parameter is passed to the function with initial values.
- And then inside the function, the new value is set for the IN OUT parameter and returned to the caller.
- The actual parameter must be a variable.

PL/SQL function

- The function must have at least one RETURN statement in the execution part.
- The RETURN clause in the function header specifies the data type of returned value.
- The block structure of a function is the same as an PL/SQL block except for the addition CREATE [OR REPLACE] FUNCTION, the parameters section, and a RETURN clause.

Examples of PL/SQL Function

- We are going to create a function that parses a string and returns a number if the string being passed is a number otherwise it returns NULL.

```
CREATE OR REPLACE FUNCTION try_parse
(
  iv_number IN VARCHAR2)
RETURN NUMBER IS
BEGIN
      RETURN TO_NUMBER(iv_number);
EXCEPTION
      WHEN OTHERS THEN
      RETURN NULL;
END;
```

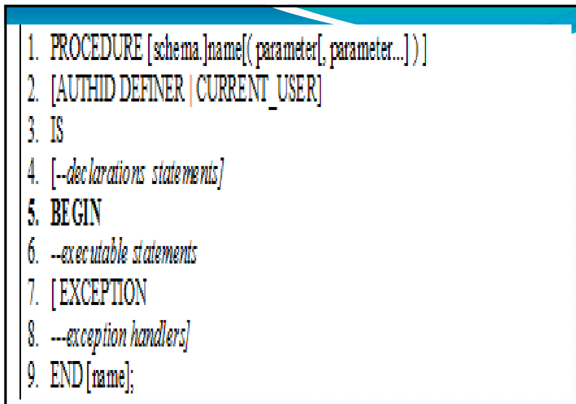
PL/SQL function

- The input parameter is iv_number that is a *varchar2* type.
- We can pass any string to the function *try_parse()*.
- We use built-in function *to_number* to convert a string into a number.
- If any exception occurs, the function will return *NULL* in the exception section of the function block.

```
SET SERVEROUTPUT ON SIZE 1000000;
DECLARE
   n_x NUMBER;
   n_y NUMBER;
   n_z NUMBER;
BEGIN
   n_x := try_parse('574');
   n_y := try_parse('12.21');
   n_z := try_parse('abcd');
   DBMS_OUTPUT.PUT_LINE(n_x);
   DBMS_OUTPUT.PUT_LINE(n_y);
   DBMS_OUTPUT.PUT_LINE(n_z);
END;
```

PL/SQL procedure

- Like a PL/SQL function, a ***PL/SQL procedure*** is a named block that performs one or more actions.
- PL/SQL procedure allows you to wrap complex business logic and reuse it.
- The following illustrates the PL/SQL procedure's syntax:



```
PROCEDURE [schema.]name[( parameter[, parameter...] ) ]
[AUTHID DEFINER | CURRENT_USER]
IS
[--declarations statements]
BEGIN
--executable statements
[ EXCEPTION
---exception handlers]
END [name];
```

- We can divide the PL/SQL procedure into two sections: header and body.
- **PL/SQL Procedure's Header**
- The section before the keyword IS is called procedures' header or procedure's signature.
- The elements in the procedure's header are listed as follows:

- Schema:
  - The optional name of the schema that own this procedure.
  - The default is the current user.
  - If you specify a different user, the current user must have privileges to create a procedure in that schema.

- Name:
  - The name of the procedure.
  - The name of the procedure like a function should be always meaningful and starting by a verb.

- Parameters:
  - The optional list of parameters.
  - Refer to the PL/SQL function for more information on parameter with different modes IN, OUT and IN OUT.

- AUTHID:
  - The optional AUHTID determines whether the procedure will execute with the privileges of the owner (DEFINER) of the procedure or the current user (CURRENT_USER).

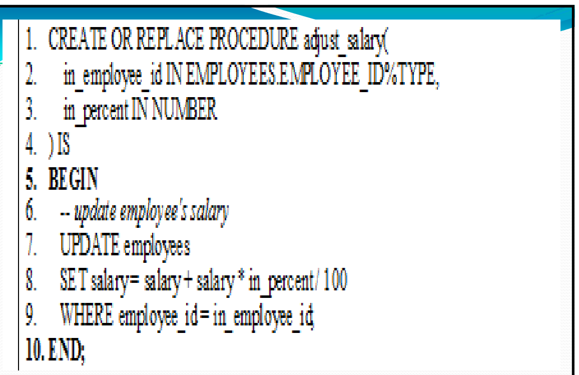
PL/SQL Procedure's Body

- Everything after the keyword IS is known as procedure's body.
- The procedure's body consists of declaration, execution and exception sections.
- The declaration and exception sections are optional.
- You must have at least one executable statement in the execution section.

- In PL/SQL procedure you still have RETURN statement.
- However unlike the RETURN statement in function that returns a value to calling program,
- RETURN statement in procedure is used only to halt the execution of procedure and return control to the caller.
- RETURN statement in procedure does not take any expression or constant.

Example of PL/SQL Procedures

- We're going to develop a procedure called *adjust_salary().*
- We'll update the salary information of employees in the table *employees* by using SQL UPDATE statement.
- Here is the PL/SQL procedure *adjust_salary()* code sample:



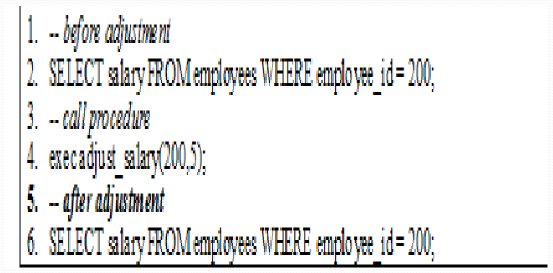
```
CREATE OR REPLACE PROCEDURE adjust_salary(
  in_employee_id IN EMPLOYEES.EMPLOYEE_ID%TYPE,
  in_percent IN NUMBER
) IS
BEGIN
 -- update employee's salary
 UPDATE employees
 SET salary = salary + salary * in_percent / 100
 WHERE employee_id = in_employee_id;
END;
```

- There are two parameters of the procedure IN_EMPLOYEE_ID and IN_PERCENT.
- This procedure will update salary information by a given percentage (IN_PERCENT) for a given employee specified by IN_EMPLOYEE_ID.
- In the procedure's body, we use SQL UPDATE statement to update salary information.
- Let's take a look how to call this procedure.

Calling PL/SQL Procedures

- A procedure can call other procedures.
- A procedure without parameters can be called directly by using keyword EXEC or EXECUTE followed by procedure's name as below:
  - EXEC procedure_name();
  - EXEC procedure_name;

- Procedure with parameters can be called by using keyword EXEC or EXECUTE followed by procedure's name and parameter list in the order corresponding to the parameters list in procedure's signature.
  - EXEC procedure_name(param1,param2…paramN);



```
-- before adjustment
SELECT salary FROM employees WHERE employee_id = 200;
-- call procedure
exec adjust_salary(200,5);
-- after adjustment
SELECT salary FROM employees WHERE employee_id = 200;
```